

推薦論文

侵入挙動の反復性を用いたボット検知方式

酒井 崇 裕^{†1} 竹 森 敬 祐^{†2}
安 藤 類 央^{†3} 西 垣 正 勝^{†4}

ボットは巧妙な手段で PC 内に潜伏するため正規プロセスとの切り分けが難しく、ボット検知のためにはボットの本質をとらえたビヘイビアの発見が求められる。ボットは外部からの制御によって動作するため、PC 内に常駐してハードからの指令を待ち受ける必要がある。このため、多くのボットにとって、初期感染の際にシステムフォルダ内に侵入し、自身を OS の自動実行リストに登録すること（以下、侵入挙動）は必須のビヘイビアとなっている。このため、環境に応じて侵入挙動と攻撃挙動を使い分けるボットであれば、実行環境を感染前の状態に戻してやることによって、侵入挙動が再び観測されると考えられる。また、単純に侵入挙動と攻撃挙動を行い続けるボットであれば、侵入挙動がつねに観測される。そこで本研究では、このボットの「侵入挙動の反復性」をボットの本質的なビヘイビアと定義し、これを利用してボットを検出する方式を提案する。

A Bot Detection Based on the Repetitiveness of Intrusion

TAKAHIRO SAKAI,^{†1} KEISUKE TAKEMORI,^{†2} RUO ANDO^{†3}
and MASAKATSU NISHIGAKI^{†4}

Due to bot's sophisticated techniques for hiding itself, it is difficult to distinguish the bot's malicious process from legitimate process. Hence it is quite essential for bot detection to find bot's inevitable behaviors. The bots are remotely controlled by commands sent through the Internet from a bot-master. This means that these bots have to stay alive themselves in PC so that they can await for further commands from the bot-master. In other words, for almost all bots, intrusion into system directory and registration themselves to auto run list are key functions which they should equip. Therefore, a clever bot, which has both intrusion and attack behaviors and separate them according to the execution environment, will exhibit again its intrusion behavior as long as its

environment is restored to pre-intrusion state. Needless to add, a naive bot, which simply iterates intrusion and attack behaviors, will always show its intrusion behavior. Therefore in this paper, we focus on this characteristic of “the repetitiveness of intrusion” as a bot's inevitable behavior and propose a bot detection scheme to utilize the characteristic.

1. はじめに

近年、ボットと呼ばれる悪質なプログラムがインターネット上に横行し、被害が増加している¹⁾。不正者は複数のボットをボットネットと呼ばれる大規模なネットワーク単位で制御し悪意を働くため、その被害も非常に大きなものとなる。また、ボットはこれまでのマルウェアに見受けられた自己顕示目的による利用のされ方から、金銭目的での利用に移り変わったといわれており²⁾、不正者はボットネットを維持するために、より緻密な手法を使ってボットを PC に潜伏させようとする。そのため、従来の検知技術ではボットを特定することは難しく、新たな検知手法の発見が望まれる。

現在、ボット対策として最も一般的な方式にパターンマッチング法³⁾がある。これは、アンチウィルスソフトで採用されている方式であり、あらかじめボットのコードパターンをシグネチャとして登録しておき、そのシグネチャと照合することでボットであるかを判定する。この方式は、単純かつ高速に処理でき、既知のボット検知に対しては非常に有効であるといえる。しかし、ボット作成のためのツールやプログラムコードがインターネット上に回っており、不正者は自由に亜種を作りだすことができるという状況⁴⁾が、ボットに様々な亜種の存在を許している。同種のボットであってもその多くはシグネチャパターンが異なる。このようにパターンが異なる未知ボットに対しては、パターンマッチング法では対応することができない。未知ボットを検出する検知方式としては、静的解析に基づくヒューリスティック法と動的

^{†1} 静岡大学大学院情報学研究科

Graduate school of Informatics, Shizuoka University

^{†2} 株式会社 KDDI 研究所

KDDI R&D Laboratories, Inc.

^{†3} 独立行政法人情報通信研究機構情報通信セキュリティ研究センター

Tractable Network Group, National Institute of Information and Communication Technology

^{†4} 静岡大学創造科学技術大学院

Graduate School of Science and Technology, Shizuoka University

本論文の内容は 2009 年 10 月のマルウェア対策研究人材育成ワークショップ 2009 にて報告され、コンピュータセキュリティ研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である。

解析に基づくビヘイビア法がある³⁾。ヒューリスティック法は、ボットによく見受けられる処理をコードの静的解析によって検知する方式である。そのため、未知ボットであってもそのコードの中から「悪意のある処理」を検出することにより、ボットとして検知することが可能である。しかし、ボットは目的によって多種多様な処理を行うため、ボットとして特徴的な処理そのものを定義することが難しい。加えて、ボットは耐解析機能として自身のファイルを暗号化・難読化するという機能を備えていることが多く、コードの解析そのものが困難である⁵⁾⁻⁷⁾。一方、ビヘイビア法は、実際にボットを実行させることによって、ボットの特徴的な振舞いをその挙動から検知する方式である。そのため、ヒューリスティック法で問題となる暗号化・難読化といった耐解析機能に関係なくボット検知を行うことができる。しかし、ボットの多様性のために、ビヘイビア法においても、やはり、ボットとしてのビヘイビアを定義することは難しい。加えて、ボットは実行環境（実行時刻、実行場所、ネットワーク環境など）によってその挙動が異なるものが多く、ボットの断定に利用可能なビヘイビアが仮にあったとしても、それがつねに観測されるとは限らない。以上のように、既存の各方式においてそれぞれ問題点が存在しており、効果的なボット検知を実現するためにはこれらの問題点を解決する必要がある。

本論文では、未知ボットを検知可能な方式であり、暗号化・難読化による耐解析機能に影響を受けないビヘイビア法に注目する。ビヘイビア法では、前述のとおり、ボットの挙動を単に受動的に観測するだけでは、ボットの本質的なビヘイビアを見い出すことは不可能に近い。そこで本論文では、「侵入機能と攻撃機能を1つの検体の中にあわせ持つタイプのボット」に焦点を絞ったうえで、ボットの実行環境をコントロールすることによって、ボット特有の挙動を能動的に引き出す方法について検討する。以下、2章で既存のビヘイビア法を概説し、課題をまとめる。3章では、侵入機能と攻撃機能を1つの検体の中にあわせ持つタイプのボットの特徴を再検討し、ハーグからの指令を受け取るためにPCに常駐するというボットの本質的な挙動を能動的に再発生させるという方法を提案する。そして、これによって観測される挙動を、ボットの「侵入挙動の反復性」というビヘイビアとして定義する。4章および5章で侵入挙動の反復性に基づく検知方式を定式化し、6章の実証実験により提案方式の効果を検証する。最後に7章で本論文をまとめる。

2. ビヘイビア法に関する既存研究

ビヘイビア法では、実行中のプロセスのファイルアクセスや通信（通信プロトコルや通信量）などを監視し、その挙動に不審な振舞いが見られた際にアラートをあげる。判定の方式

によって、ホワイトリスト型とブラックリスト型に大別される。

ホワイトリスト型の方式は、正規プロセスの「通常の振舞い」をルール化し、プロセスの挙動が通常の振舞いを逸脱する際にこれを異常として検知する^{8),9)}。しかし、多種多様な正規プロセスの「通常の振舞い」を完全に正しくルール化することは非常に困難である。

一方、ブラックリスト型の方式は、ボットによく見受けられる「異常な振舞い」をルール化し、プロセスの挙動がこれに触れた際にこれを異常として検知する。しかしボットは、ファイルの破棄や強制シャットダウン、大規模感染活動などといった表立った行動を行うウイルスやワームと異なり、感染先PCの中に潜むため、挙動検出が難しい。また、指令サーバとの通信や他のPCへの攻撃（DoS攻撃、スパム発信）・感染（エクスプロイト、ポートスキャン）を行う際においても、正規プロトコルを装うとともに通信量を制御することによって正規の通信になりすます。このため、ブラックリスト型方式の効果を高めるためには、真に「ボットらしい振舞い」をビヘイビアとして規定することが重要となる。

松本らは、ワームが感染時に自分自身の実行形式ファイル（自己ファイル）を再度読み出すという挙動に注目し、「実行プロセスによる自己ファイルのREAD」をワームの特徴的なビヘイビアとして定義した¹⁰⁾。著者らは、松本らの方式をボット検知に適用し、自己ファイルREADがボットにとっても特徴的なビヘイビアである（ボットがPC内に自身を侵入させる際にも自己ファイルREADの挙動が検出される）ことを確認している¹¹⁾。しかし、自己ファイルREADは正規プログラムであるインストーラやアンインストーラの実行の際にも観測される挙動であり、これらの切り分けが必要となる。

福島らの方式¹²⁾では、インストーラ/アンインストーラが一般的にアンインストール情報の登録/削除を行うことに注目し、インストーラ/アンインストーラの識別を行っている。しかし、現在インターネットで入手可能なフリーソフトやシェアウェアなどは、アンインストール情報を適正に管理しているとは限らない。実際マルウェアは、このような身元の分からないソフトウェアに混在してPC内に侵入してくることが多いため、「アンインストール情報の登録」というビヘイビアは万全とはいえない。

本論文では、ボットにとって本質的であり、かつ、正規プログラムとの誤検知のない振舞いを導出し、これをブラックリスト型のビヘイビア検知方式に適用することを目的とする。

3. ボットの特徴

ボットの挙動は多種多様であるが、その一連の挙動は侵入挙動と攻撃挙動の2つに大別することができると考えられる。侵入挙動とは、ボットが初期感染時にPCに潜りこむ際に行

う、自身の潜伏環境を整えるための挙動であり、OS 上のファイルおよびレジストリの書き換えや、自分自身の実行ファイルを潜伏先フォルダ^{*1}内にコピーするなどの振舞いを示す。攻撃挙動は、ボットが PC に潜伏した後、外部の指令サーバなどから受信した指令に従い、潜伏先フォルダ内で行う攻撃活動に関する振舞いを示す。

侵入と攻撃の 2 つの機能は、ボットが自身の目的を達成するために不可欠なものである。よって、ボットはこの侵入・攻撃の両機能を単一の検体の中に持っているか、または、侵入機能のみを持つ検体と攻撃機能のみを持つ検体からなる複数のボット群が全体で 1 つのボットとして動く（たとえばダウンロードタイプのボットなど）ことになる。本研究の現段階では、このうち、前者の「侵入機能と攻撃機能を 1 つの検体の中にあわせ持つタイプのボット」を検知対象とする。以降、単に「ボット」と記した場合は、本論文の検知対象のボットを指す。

ボットは自身を PC 内に潜伏させた後は、外部からの指令によって制御されるため、その攻撃の種類やタイミングが不定となり、攻撃挙動のみからボットを捕えることは難しい。しかし、ボットが外部からの制御によって動作するためには、PC 内に常駐して指令サーバからの指令を待ち受けるか、または、タイマで自分自身を定期的に起動して指令サーバに問い合わせるようにする必要がある。そのため、ボットにとって侵入の際に自身を OS の自動実行リスト（レジストリ、スタートアップフォルダ、サービスプロセスなど）に登録するというビヘイビアが必須となってくると考えられる^{*2}。本論文では、以下、「侵入挙動」という言葉は、自動実行リストへの自分自身の登録を指すこととする。なお、前述のとおり、ボットは自分自身を潜伏先フォルダ（主にシステムフォルダ）に複製（または移動）したうえで、その複製（または移動後の検体）を自動実行リストへ登録するものが多い。したがって、ファイルの複製（または移動）の際に観測される「ファイル生成」のイベントもボットの侵入挙動に含めるものとする。

侵入挙動はボットにとって「必須」のビヘイビアではあるが、これは正規のインストーラなどにも見られる挙動であるため、侵入挙動をボットにおいてのみ観測される「特有」のビヘイ

ビアとして定義することはできない。そのため、さらなるボットの特徴を検討する必要がある。

侵入と攻撃の 2 つの機能を単一の検体の中に持っているボットは、PC 内で初めて実行されたとき（初期感染時）には侵入挙動を行い、侵入後は攻撃挙動に移るというように、段階的に挙動を変化させることが一般的である。これは逆にいえば、侵入後のボットの実行環境を初期感染前の状態に戻した場合には、ボットは再び PC への侵入を図り、侵入挙動が観測されることを意味する。なお、単純に侵入挙動と攻撃挙動を行い続けるようなボットの場合は、実行環境を初期感染前の状態に戻さなくてもつねに侵入挙動が観測される。この「侵入挙動が繰り返される」というビヘイビアは、インストーラなどの正規プロセスでは通常は起こりえない挙動である。よって著者らは、侵入機能と攻撃機能を 1 つの検体の中にあわせ持つタイプのボットが有する本質的なビヘイビア（ボットにとって必須であり、かつ、ボットに特有のビヘイビア）であると考えた。

本論文では、この侵入挙動の繰返しを「侵入挙動の反復性」と定義し、このビヘイビアを利用することによって侵入機能と攻撃機能を 1 つの検体の中にあわせ持つタイプのボットの検知を行う方式を提案する。ここで、ボットの侵入挙動の反復性を検査するために、ボットの実行環境を初期状態に戻すことに注意されたい。すなわち、ボットの挙動を単に受動的に観測するのではなく、ボットの実行環境をコントロールすることによって、ボット特有のビヘイビアを能動的に引き出している。挙動を受動的に観測する方法の場合は、ボットのあらゆる挙動をつねに監視し続ける必要がある。これに対し、本方式では、実行環境を初期感染前の状態に戻したうえでボットを実行させてやれば、そのタイミングで当該ビヘイビアが現れるか否かを検査することが可能である。よって、受動的なビヘイビアの監視と比べ、ビヘイビアを能動的に引き出す本方式はボット検知が容易であるというメリットも存在する。

4. 侵入挙動の反復性

本章では、侵入挙動の反復性に関するボットと正規プログラムの動作の違いを確認する。4.1 節ではボットの挙動について、4.2 節では正規プロセスの挙動についてそれぞれ論じ、「侵入挙動の反復性」というビヘイビアによって侵入機能と攻撃機能を 1 つの検体の中にあわせ持つタイプのボットが誤検知なく検出できるか否かを検討する。

4.1 ボットにおける侵入挙動の反復性

ボットを、その侵入と攻撃の挙動に着目して分類すると、侵入挙動と攻撃挙動を使い分ける高機能なボットと、つねに両挙動を行う低機能なボットの 2 つのタイプに分けられる。以下に、それぞれのタイプに対して侵入挙動の反復性がどのように現れるか説明する。

*1 Windows においては、システムフォルダに潜伏することがほとんどである。これは、(i) システムフォルダ内には多くの実行ファイルや DLL ファイルが含まれているので、ボットの実行プログラムが追加されても気付かれにくい、(ii) 一般ユーザはシステムフォルダ内のファイルを把握していないため、ボットの実行プログラムが追加されても分からない、などの理由によると考えられている。

*2 ワームにおいては、CodeRed¹³⁾ のように自動実行リストへの登録はせずにメモリに常駐し続けるタイプも存在する。しかし、ボットネットを構成しているゾンビ PC の多くは一般ユーザの PC であり、一般ユーザは任意のタイミングで PC をシャットダウン/再起動する。そのため、自身を自動実行リストへ登録させておかなければボットは PC 内に常駐し続けることができない。

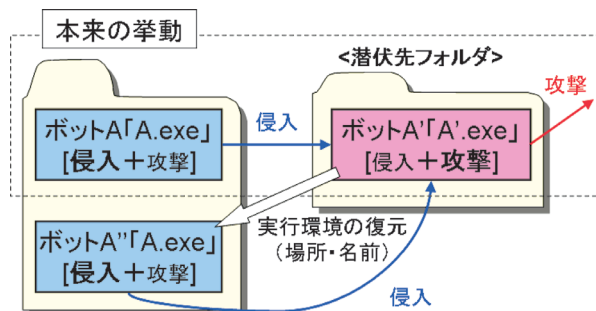


図 1 高機能ボットにおける侵入挙動の反復性
Fig. 1 Repetitiveness of intrusion by a clever bot.

●高機能ボット

潜伏先フォルダに侵入した高機能ボット A を、意図的に初期感染前の実行環境に戻して実行させた場合の、ボットの挙動を図 1 に示す。

ボット A を起動して初期感染させた場合の挙動には以下が含まれる。

- 潜伏先フォルダに自分自身の実行ファイルを複製する。なお、潜伏先フォルダのボットには、A とは異なるファイル名（図 1 の例では A'）が付けられることが多い。
- 潜伏先フォルダ内のボット A' を、OS の自動実行リスト（レジストリ、スタートアップフォルダ、サービスプロセスなど）に登録する。
- 潜伏先フォルダ内のボット A' を起動する。A' は攻撃挙動を行う。

一方、潜伏先フォルダに複製されたボット A' を起動させた場合は、ボットは侵入挙動を行わず、直接、攻撃挙動を開始する。

このように高機能ボットは、侵入機能と攻撃機能をあわせ持っており、実行環境に応じて自らの挙動を変化させている¹⁴⁾。これはすなわち、潜伏先フォルダ内に複製されたボット A' も、実行環境を感染初期の状態に戻したうえで実行した場合には、侵入挙動が再び観測されることを意味する。

本論文では、上記の侵入挙動のうち、まずは実行場所の変化とファイル名の変化に注目し、これらを「挙動を変化させる実行環境の要因」として定義する。ボット A' の実行場所を意図的にボット A のフォルダに戻した場合、また、ボット A' のファイル名を意図的にボット A のファイル名に戻した場合に、実行環境を戻されたボット A' は自身が初期感染前の状態であると認識し、再び侵入挙動を行うと予想される。

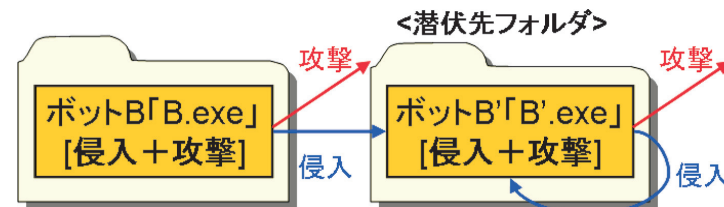


図 2 低機能ボットにおける侵入挙動の反復性
Fig. 2 Repetitiveness of intrusion by a naive bot.

●低機能ボット

低機能ボット B は、つねに侵入・攻撃の両挙動を行うため、図 2 のような挙動になると考えられる。

ボット B を起動して初期感染させた場合の挙動には以下が含まれる。

- 潜伏先フォルダに自分自身の実行ファイルを複製する。なお、潜伏先フォルダのボットには、B とは異なるファイル名（図 2 の例では B'）が付けられることが多い。
- 潜伏先フォルダ内のボット B' を、OS の自動実行リスト（レジストリ、スタートアップフォルダ、サービスプロセスなど）に登録する。
- B' を侵入させた後、B は攻撃挙動を行う。

一方、潜伏先フォルダに複製されたボット B' を起動した場合も、前述したボット B と同様の挙動を行う。ただし、ボット B の場合はボット B' の生成が成功したのに対し、ボット B' の場合は潜伏先にすでに自分自身（ボット B'）が存在するため、ファイルの生成は失敗する（実際に B' の生成が行われることはない）。

このようにボット B は、潜伏先フォルダ内に複製されたボット B' の実行環境を意図的に操作しなくても、侵入挙動の反復が観測できると予想される。

4.2 正規プログラムにおける挙動の反復性

正規プログラムであるインストーラの中には任意のフォルダに実行ファイルを生成後、自動実行リストへの登録を行うものが存在するため、これらの正規プロセスの挙動はボットの侵入挙動との区別が難しい。インストーラを例にとって説明する（図 3）。

インストーラ C を起動した場合の挙動には以下が含まれる。

- インストール先フォルダにアプリケーションソフト D を生成する。一般的に、インストーラ C とアプリケーションソフト D のファイル名は異なる。
- アプリケーションソフト D を、OS の自動実行リスト（レジストリ、スタートアップ

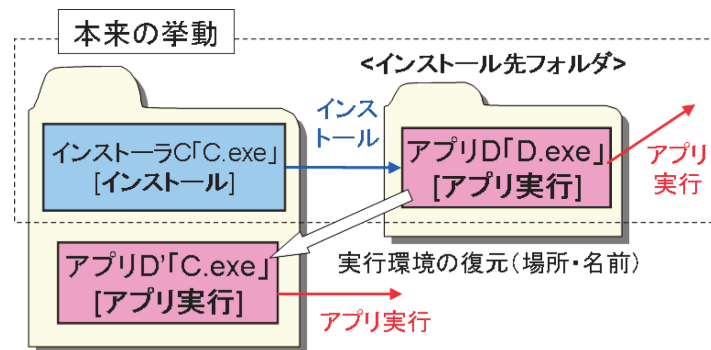


図3 正規プログラムにおける侵入挙動の反復性
Fig.3 Repetitiveness of intrusion by a legitimate program.

フォルダ、サービスプロセスなど)に登録する。

- アプリケーションソフト D を起動する。アプリケーションソフトは、目的に応じた動作を行う（たとえば、エディタソフトであれば新規文書の作成画面が表示される）。一方、アプリケーションソフト D を起動させた場合は、アプリケーションソフトの目的に応じた動作のみが観測される。

このように、インストーラにおいてもボットと同様、インストーラ C の挙動と C によって生成されたアプリケーションソフト D の挙動は異なる。しかし、ここには、侵入挙動によって生成されたボット（図1における A' や図2における B'）が侵入機能（と攻撃機能の両者）を引き継いでいるのに対し、インストールによって生成されたアプリケーションソフト D はインストール機能を持ちあわせていないという決定的な違いが存在している。このため、アプリケーションソフト D の実行場所を意図的にインストーラ C のフォルダに戻したり、アプリケーションソフト D のファイル名を意図的にインストーラ C のファイル名に変更したりしたとしても、インストーラ C の実行環境に戻されたアプリケーションソフト D' の起動においてインストールの挙動が再び観測されることは起こりえない。

正規プログラムにおいては、アプリケーション（図3の例では D）の実行中にユーザーの意思で自身の常駐/非常駐の設定を切り替えることができるものも存在する。このようなアプリケーションにおいては、アプリケーションソフト D または D' を実行した際にユーザーがアプリケーションソフトの常駐化の操作を行った場合には、自分自身を自動実行リストに登録する挙動が観測される。しかし、これは、ボットのように無条件に自分自身を自動実行

リストに登録する挙動とは異なる。よって、自動実行リスト登録についてはユーザーの操作の有無をあわせて検査する、または、検査の際にはユーザーの操作を禁止することによって誤検知を抑制可能である。

5. 提案方式

本章では、「侵入挙動の反復性」をビヘイビアとしたボット検知方式のアルゴリズムを説明する。本方式の概要を図4に示す。

1. 実行中の全プロセスを監視し、いずれかのプロセスにおいて侵入挙動（実行ファイルの生成、および、生成された実行ファイルの自動実行リストへの登録）が観測された時点で、そのイベントを起点として、そのプロセスに対して2.の検査を行う。このとき、侵入挙動を行ったプロセスの実行ファイルを α 、 α によって生成された実行ファイルを β と呼ぶ。
2. β を α と同様の実行環境に復元することによって、検査用の実行ファイル γ を生成する。4.1節で示したように、本論文では、実行場所の変化とファイル名の変化を「挙動を変化させる実行環境の要因」として定義した。よって、今回の提案方式においては、 β を α のフォルダに移動し、 β のファイル名を α と同じファイル名に変更したものが γ となる。なお、 γ の生成の際には、 α の侵入挙動によって自動実行リストに登録された実行ファイルについてはそのエントリがリストから削除される。
3. γ を実行させた際に再び侵入挙動（実行ファイルの自動実行リストへの登録）が行われたかどうかを検査する。この結果、 γ による侵入挙動が観測された場合には α および β をボットとして検出する。なお、 γ の実行はユーザーからは隔離されており、ユーザーが γ を操作することはない。

高機能ボットに関しては、図1におけるボット A' が図4における γ に対応するため、本方式によってボット検知が可能であることが容易に理解できる。つまり、侵入したボット A' (β) を侵入前のボット A (α) と同様の実行環境に戻したボット A'' (γ) の侵入挙動の有無を監視することでボットであるかを検知する。低機能ボットに関しては、侵入挙動を繰り返すボット（図2における B'）は図4の β に対応する。つまり、低機能ボットを検査する場合には、本来であれば β における侵入挙動の有無を監視すればよい。しかし本論文では、検知アルゴリズムを統一するため、低機能ボットであっても β から γ を生成し、 γ における侵入挙動を検査することとする。低機能ボットはどのような環境であっても侵入挙動を行うため、 γ においても β と同様の侵入挙動を観測可能である。一方、正規プログラムの場合

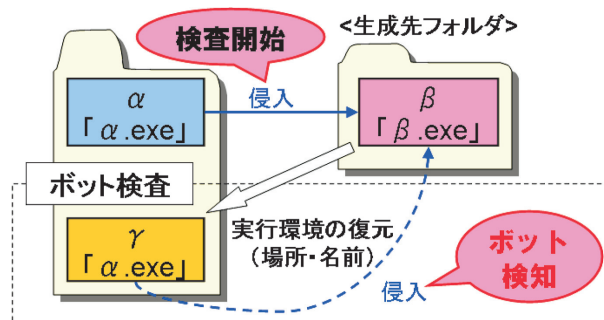


図4 提案方式の概要

Fig. 4 Overview of proposed method.

には、図3におけるアプリケーションソフトD'が図4における γ に対応する。前述のとおり、基本的にユーザの操作なしでD'が自身を自動実行リストに登録するという動作が発生することはないため、ユーザから隔離された γ の実行の際に侵入挙動が観測されることはなく、本検知方式がアプリケーションソフトD'をボットと誤検知する可能性は無視できると期待される。以上より、本検知アルゴリズムが、4章で説明した「侵入挙動の反復性」を正しくとらえることができる手順となっていることが分かる。

ここで、本検知方式では α の侵入挙動を検査開始としているが、これは、 β から γ を生成するうえで（ β の実行環境を α の実行環境に戻すにあたり）、 α の実行環境が必要であるからである。通常、 β がインストールされてしまった後の時点で α の実行環境を知るすべはないため、PCにすでに侵入してしまった後のボットに対しては本方式の適用が難しい。このため、本方式では、 α が侵入挙動を行う時点、つまりPC内に新たにボットが侵入する時点トリガとして検査を行う方法をとっている。ただし、6.2節で述べるように、自身（ β ）が生成先フォルダに存在しなければ再び侵入挙動を行うボットが多数確認されているため、 β から γ を生成する際に確実に α の実行環境を再現しなくても侵入挙動の反復性が観測される場合もある。このようなボットの場合は、PCにすでに潜伏しているものであっても（ α の環境が分からなくても）本方式によって検査が可能である。

なお、本方式を実装するためには、侵入挙動が行われた瞬間を検出する必要があるが、これはOSのシステムコールAPIをフックし、自動実行リストへの登録の際に利用されるファイルやレジストリアクセスなどをリアルタイムで監視することによって可能となる。

6. 検 証

6.1 検 証 方 法

本方式によるボットの検知および正規プロセスの誤検知について検証する。検証にあたってはボットをリアルタイムで検知する必要はないため、ファイルおよびレジストリアクセスに関するAPIをフックする代わりに、プロセスのファイルおよびレジストリアクセスを監視可能なモニタツールであるProcMon¹⁵⁾を用いて自動実行リスト登録とファイル生成のイベントを抽出することとした。ここで、自動実行リストとしては以下のものを想定した。

●各種レジストリ

- Run キー：HKLM¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Run
HKCU¥Software¥Microsoft¥Windows¥CurrentVersion¥Run など
- サービス：HKLM¥System¥CurrentControlSet¥Services
- その他自動実行に関わるレジストリ：
自動実行リスト上のプログラムを一覧表示するモニタツールであるAutoruns¹⁶⁾の持つレジストリリストに含まれるレジストリ

●スタートアップフォルダ

●タスクスケジューラ

また、 β から γ を生成するにあたっては、 α が自動実行リストに登録したエントリを削除する必要があるが、今回は、仮想マシンを用い、PC全体を α の実行前の実行環境に戻すことによって、これを実現することとした。

本実験は、物理的に隔離されたネットワーク上で行った。実験に使用した仮想マシンは、仮想マシンソフトがVMWare Workstation6、仮想マシン上で動作させたゲストOSがWindows XP Professional SP2である。

6.2 検 知 実 験

本方式によって実際にボットが検知可能であるか実験を行った。検証実験に使用したボットは、研究用データセットCCC DATASET 2009¹⁷⁾のハッシュ値に該当する検体および、著者らの研究室で独自に収集したボットと疑いのある検体の合計149体である。ここで、ボットはその機能が多種多様であり、他のマルウェアとの境界が曖昧である。本実験においては、起動後に指令サーバと思わしき相手との通信と見受けられる挙動が観測されている検体を「ボットと疑いのある検体」として利用したが、ボット以外のマルウェアが一部含まれている可能性がある。

表 1 検知実験結果

Table 1 Experimental results for bot detection.

ボット検体	α 実行時の 自動実行 リスト登録	侵入挙動の反復			判定	検知率
		場所を戻し た際に発生	名前を戻し た際に発生	常時 発生		
A グループ (3/149 体)	○	○	○	×	○(検知)	100% (107/107 体)
B グループ (56/149 体)	○	○	×	×	○(検知)	
C グループ (1/149 体)	○	×	○	×	○(検知)	
D グループ (47/149 体)	○	×	×	○	○(検知)	
E グループ (42/149 体)	×	—	—	—	— (検知 対象外)	— (42 体)

表 1 に検知実験の結果を示す．表中の「 α 実行時の自動実行リスト登録」の列には，検体を実行した際に実行ファイルを自動実行リストに登録した検体に ○ が付けられている．本方式は，自動実行リスト登録が観測された検体に対して，侵入挙動の反復の有無を検査する．表中の「侵入挙動の反復」の列には，侵入挙動が反復される要因となった実行環境を示している（ β を α のフォルダに戻した際に侵入挙動の反復が観測された検体に対しては「場所を戻した際に発生」に ○ が， β の名前を α に戻した際に侵入挙動の反復が観測された検体に対しては「名前を戻した際に発生」に ○ が， β の侵入挙動の反復がつねに観測された検体に対しては「常時発生」に ○ が付けられている）．検体はタイプ別にグループ化して集計し，検知率を算出した．

ここで，実行環境（場所・名前）を侵入前の状態に戻した場合にのみ侵入挙動が再び観測されたグループ A～C の検体は，4.1 節で説明した高機能ボットであると考えられる．また，つねに侵入挙動が観測されたグループ D の検体は，4.1 節で説明した低機能ボットであると考えられる．なお，自動実行リストへの登録が観測されなかったグループ E の検体は，侵入挙動と攻撃挙動を別々の個体によって実装している検体であると推測され，たとえばダウンロードタイプのボットである可能性が考えられる．本論文では「侵入機能と攻撃機能を 1 つの検体の中にあわせ持つタイプのボット」を検査対象としているため，ダウンロードタイプのボットは本方式の対象外となる．

表 2 誤検知実験結果

Table 2 Experimental results for false detection.

正規プログラム	α 実行時の 自動実行リ スト登録	γ 実行時の 自動実行リ スト登録	判定	誤検知率
グループ F (7/15 個)	×	×	×	0% (0/15 個)
グループ G (8/15 個)	○	×	×	

表 1 に示すように，本論文の検査対象となるグループ A～D に属するすべての検体において，侵入挙動の反復が観測され，本方式での検出が可能であることが確認できた．

ここで，高機能ボットにおける侵入挙動の反復性が再現された要因としては，グループ A または B のように「場所」によるものがほとんどであった．これに関して詳しく調査したところ，グループ A，B のすべての検体において， β を α のフォルダに戻さなくても， β を生成先フォルダ以外の場所に移動すれば侵入挙動の反復が確認された．また，グループ D（低機能ボット）も，前述のとおりつねに侵入挙動（と攻撃挙動）を繰り返すため， β がどこに置かれたとしても侵入挙動の反復が観測される．このように，現存する多くのボット（表 1 におけるグループ A，B，D に属するボット）については，すでに PC に侵入してしまっている状態であっても（ α の実行環境が不明であっても）， β の存在するフォルダ以外の場所に γ を生成することによって， β がボットであるか否かが判定することが可能である．

以上のように，侵入挙動と攻撃挙動をあわせ持つタイプのボットに対しては，本方式が有効であることを示すことができた．

6.3 誤検知実験

本方式によって正規のプログラムがボットとして誤検知されることがないかを調べる実験を行った．今回の実験では，一般的なユーザが日常的に利用すると予想されるアプリケーションのインストーラ 15 個を正規のプログラムとして採用した．具体的には，Microsoft Office2003，ブラウザ（Firefox など），Adobe Reader，そのほかにフリーソフトとしてカレンダー管理ツール（Rainlendar¹⁸⁾）やクイックバー管理ツール（ExtendQuickBar¹⁹⁾）など多岐にわたる正規プログラムについて評価している．表 2 に誤検知実験の結果を示す．表中の「 α 実行時の自動実行リスト登録」は，インストーラ（図 4 における α ）を実行した際

に自動実行リスト登録の挙動が観測された場合に○が付けられる。「 γ 実行時の自動実行リスト登録」は、インストーラによって生成された実行ファイル(図4における β)をインストーラの実行環境に戻したうえで実行(図4における γ)した際に自動実行リスト登録の挙動が観測された場合に○が付けられる。誤検知実験においても、そのタイプ別にプログラムを集計し、誤検知率を算出した。

グループFについては、MS Office2003, Firefox, Adobe Readerなどのインストーラが含まれており、これらはそもそもインストール時に実行ファイルを自動実行リストに登録することがなかった。5章に記したように、提案方式は、実行ファイルの(生成と)自動実行リストへの登録を侵入挙動と定義しており、それが観測された時点でボット検査を開始する。よって、これらのインストーラの実行においては、ボット検査が開始されなかった。

グループGについては、Skype, Rainlendar, ExtendQuickBarなどのインストーラが含まれる。これらは、まずインストーラ(図4における α)を起動してインストールを行った際に、実行ファイル(図4における β)の生成と生成された実行ファイルの自動実行リストへの登録が観測された。そして、インストールによって生成された実行ファイル(β)をインストーラ(α)の実行環境に戻した実行ファイル(図4における γ)を実行した際には、実行ファイルの生成と自動実行リスト登録の挙動はともに観測されなかった^{*1}。

以上のように、本方式は正規プログラムの誤検知が発生しないことを示すことができた。

7. ま と め

本論文では、ボットが侵入機能と攻撃機能をあわせ持つという特徴を通じて、「侵入挙動の反復性」という挙動に注目したボット検知方式の提案および検討を行った。ボットの実行環境をコントロールして、侵入挙動の反復というビヘイビアを能動的に引き出すことが提案方式の特長である。

監視ツールを用いた基礎実験から、本検知方式を用いることで、正規アプリケーションソフト(インストーラ)を誤検知することなく、侵入機能と攻撃機能をあわせ持つボットを100%の精度で検知できることが確認された。これにより、「侵入挙動の反復性」を侵入機能と攻撃機能をあわせ持つボットにとって本質的なビヘイビア(ボットとして必須であり、か

*1 γ を実行した際に、ユーザがアプリケーションの常駐に関する設定変更の操作を行った場合には、自分自身(γ)を自動実行リストに登録するという挙動が観測された。本検知方式では、 γ の実行中にユーザが γ を操作することはないため誤検知には至らない。なお、ユーザが操作した場合であっても、 γ がさらに実行ファイルを生成することはなかった。

つ、特有なビヘイビア)として利用するというアプローチの妥当性が示されたと考えている。

今後は、侵入挙動およびその反復性の定義を改良することで本方式の対応可能範囲を広げ高めるとともに、侵入機能と攻撃機能をそれぞれ単体で有するタイプのボットも検知可能なビヘイビアを導出していきたい。

謝辞 本研究は一部、(財)セコム科学技術振興財団の研究助成を受けた。ここに深く謝意を表する。

参 考 文 献

- 1) サイバークリーンセンター：平成19年度サイバークリーンセンター(CCC)活動報告。 <https://www.ccc.go.jp/report/h19ccc-report.pdf>
- 2) Friess, N., Aycok, J. and Vogt, R.: Black Market Botnets, *The Expanded MIT Spam Conference 2008* (2008.3).
- 3) 情報処理推進機構：未知ウイルス検出技術に関する調査。
<http://www.ipa.go.jp/security/fy15/reports/uvd/index.html>
- 4) サイバークリーンセンター：ボットとは。<https://www.ccc.go.jp/bot/index.html>
- 5) OllyBonE. <http://www.joestewart.org/ollybone/>
- 6) Kang, M.G., Poosankam, P. and Yin, H.: Renovo: A hidden code extractor for packed executables, *Proc. 2007 ACM Workshop on Recurring Malcode (WORM '07)*, pp.46-53 (2007.7).
- 7) 岩村 誠, 伊藤光恭, 村岡洋一：コンパイラ出力コードモデルの尤度に基づくアンパッキング手法, コンピュータセキュリティシンポジウム2008 論文集(併設:MWS)(2008.10).
- 8) Binkley, J.R. and Singh, S.: An Algorithm for Anomaly-based Botnet Detection, *Computer Science, PSU, USENIX SRUTI '06 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet* (2006).
- 9) Binkley, J.R.: Anomaly-based Botnet Server Detection, *Computer Science, PSU, FLOCON CERT/SEI, Vancouver, WA* (2006.10).
- 10) 松本隆明, 鈴木功一, 高見知寛, 馬場達也, 前田秀介, 水野忠則, 西垣正勝：自己ファイルREADの検出による未知ワームの検知方式, 情報処理学会論文誌, Vol.48, No.9, pp.3174-3182 (2007).
- 11) 酒井崇裕, 長谷 巧, 竹森敬祐, 西垣正勝：自己ファイルREAD/DELETEの検出によるボット検知の可能性に関する一検討, コンピュータセキュリティシンポジウム2008 論文集(併設:MWS)(2008.10).
- 12) 福島祥郎, 境 顕宏, 堀 良彰, 櫻井幸一：プロセスの振る舞いに着目したマルウェア検知手法の実装とその評価, DICOMO2009 論文集, pp.1261-1268 (2009).
- 13) Trend Micro: CodeRed. <http://www.trendmicro.co.jp/vinfo/virusencyclo/>

default5.asp?VName=CODERED.A

- 14) 酒井崇裕, 竹森敬祐, 安藤類央, 西垣正勝: 実行環境による挙動変化を用いたボット検出方式の提案, 情報処理学会研究報告, 2009-CSEC-46-37 (2009.7).
- 15) Microsoft TechNet: ProcMon.
<http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx>
- 16) Microsoft TechNet: Autoruns.
<http://technet.microsoft.com/ja-jp/sysinternals/bb963902.aspx>
- 17) 畑田充弘ほか: 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* (2009.10).
- 18) Rainy: Rainlender. <http://www.rainlendar.net/cms/index.php>
- 19) mebiusbox software: ExtendQuickBar.
http://mebiusbox.crap.jp/software_extend_quick_bar.html

(平成 21 年 11 月 30 日受付)

(平成 22 年 6 月 3 日採録)

推 薦 文

本論文は, ボットの本質的な振舞いと見なせる「侵入挙動の反復性」に着目した検知方式を提案している. ここで侵入挙動とは, システムフォルダ内に侵入し, 自身を OS の実行リストに登録する, という, ボットにとっては非常に重要なアクションである. さらに, CCC DATASET 2009 のマルウェア検体を用いた検知実験, 正規プログラムを用いた誤検知実験を行い, 本方式の有効性を評価している. ボットは巧妙な手段で潜伏するため正規プロセスとの区別が困難であるが, 提案方式はボットの本質をとらえたものであり, その有効性は高いものであると判断し, 推薦する.

(コンピュータセキュリティ研究会主査 菊池浩明)



酒井 崇裕

2007 年静岡大学情報学部情報科学科卒業. 現在, 同大学大学院修士課程. 情報セキュリティに関する研究に従事.



竹森 敬祐 (正会員)

1994 年慶應義塾大学理工学部電気工学科卒業. 1996 年同大学大学院修士課程修了. 同年 KDD 入社. 2004 年慶應義塾大学大学院博士課程修了. 現在, KDDI 研究所に勤務. ネットワークセキュリティ, モバイルセキュリティに関する研究に従事. 博士 (工学). 2002 年度電子情報通信学会学術奨励賞, 2006 年 Interop グランプリ, 2007 年 DICOMO 最優秀論文賞, 2009 年 MWS 優秀論文発表賞, 2010 年山下記念研究賞各受賞. IEEE, 電子情報通信学会各会員.



安藤 類央 (正会員)

2002 年 6 月慶應義塾大学大学院政策メディア研究科前期博士課程修了 (修士 政策・メディア). 2006 年 3 月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 (博士 政策・メディア). 同年情報通信研究機構情報通信セキュリティ研究センターに勤務. ネットワークセキュリティ, 仮想化システムに関する研究に従事. 博士 (政策・メディア).



西垣 正勝 (正会員)

1990 年静岡大学工学部光電機械工学科卒業. 1992 年同大学大学院修士課程修了. 1995 年同博士課程修了. 日本学術振興会特別研究員 (PD) を経て, 1996 年静岡大学情報学部助手. 1999 年同講師, 2001 年同助教授. 2006 年より同創造科学技術大学院助教授. 2007 年より准教授. 博士 (工学). 情報セキュリティ, ニューラルネットワーク, 回路シミュレーション等に関する研究に従事.