侵入挙動の反復性によるボット検知方式

酒井崇裕[†1]　竹森敬祐[†2]　安藤類央[†3]　西垣正勝[†4]

[†1]静岡大学大学院情報学研究科，〒432-8011 浜松市中区城北 3-5-1
[†2]株式会社 KDDI 研究所，〒356-8502 埼玉県ふじみ野市大原 2-1-15
[†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
〒184-8795 東京都小金井市貫井北町 4-2-1
[†4]静岡大学創造科学技術大学院，〒432-8011 浜松市中区城北 3-5-1

あらまし　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捉えたビヘイビアの発見が求められる．ほとんどのボットにとって，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）は非常に重要なアクションとなっている．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，実行環境を感染初期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．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，侵入挙動が常に観測される．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]　Masakatsu Nishigaki[†4]

[†1]Graduate school of Informatics, Shizuoka University,
3-5-1 Johoku, Naka, Hamamatsu, 432-8011 Japan
[†2] KDDI R&D Laboratories, Inc. 2-1-15 Ohara, Fujimino, Saitama, 356-8502 Japan
[†3]National Institute of Information and Communication Technology, Tracable Network Group,
4-2-1 ,Nukui-Kitamachi, Koganei,Tokyo 184-8795 Japan
[†4]Graduate School of Science and Technology, Shizuoka University,
3-5-1 Johoku, Naka, Hamamatsu, 432-8011 Japan

Abstract 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. For almost all bots, intrusion into system directory and registration themselves to auto run list are key function which they should equip to stay alive themselves in PC. 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を横行し，被害が増大している[1]．その対策として，これまでに様々なボット検知手法が提案されてきているが，著者らは，未知のボットを効果的に検知することが可能であるという観点から，ビヘイビアブロッキング法[2]に注目している．ビヘイビアブロッキング法では，システム上で動作しているプロセスの動きを監視し，ボットによく見受けられる挙動を検出することによってボット検知を行う．しかしながらボットは，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，ビヘイビアそのものの検出が難しい．また，他の PC に攻撃（DoS 攻撃，スパム発信）や感染（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，ボットと正規プログラムを切り分けることは一般的に困難である．すなわち，ビヘイビアブロッキングによってボットを検知するためには，ボットの本質を捉えたビヘイビアを定義することが非常に肝要となる．

ボットの一連の挙動は，侵入挙動と攻撃挙動の 2 つに分類することができると考えられる．侵入挙動は，ボットが始めにPCに潜りこむ際に，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ビヘイビアを示す．攻撃挙動は，PCに侵入・潜伏後，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ビヘイビアを示す．ここで，ボットにとってPC内に潜伏・常駐し続けることが非常に重要であるため，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（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）に登録するという一連の挙動は必須のアクションであると考えられる．本稿では，以下，「侵入挙動」という言葉は特にこのアクションのことを指すこととする．侵入挙動はボットにとって本質的なアクションであるとともに，任意のタイミングで実行される攻撃挙動と比べ観測が容易であるといえる．そこで，侵入挙動に注目してボットの特徴的なビヘイビアを検討する．

ボットは侵入挙動と攻撃挙動を併せ持っているため，PC 内で初めて実行された時には侵入挙動を行い，侵入後に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[3]．これはすなわち，実行環境に応じて侵入挙動と攻撃挙動を使い分けるボットにおいては，実行環境を感染初期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ことを意味する．また，常に侵入挙動と攻撃挙動を行い続けるような単純なボットの場合は，実行環境を変化させずとも常に侵入挙動が観測される．著者らは，この「侵入挙動が繰り返されるというビヘイビア」が，ボット特有のビヘイビアであると考えた．本稿では，この侵入挙動の繰り返しを「侵入挙動の反復性」と定義し，この性質を利用することによってボット検出を行う方式を提案する．

2　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．2.1 節ではボットの挙動について，2.2 節では正規プロセスの挙動について論じ，ボットを検出しえる特徴的なビヘイビアが存在するかどうかを検討する．両者に決定的な違いが存在すれば，誤検知無しでボットを検出できる可能性が示唆される．

2.1　ボットにおける侵入挙動の反復性

ボットを，そ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，常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

*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ない，(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，(iii) システムリソースを使用しやすい，などの理由によると考えられている．

る.

● 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，意図的に侵入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ボットの挙動には以下が含まれる.

- 潜伏先フォルダに自分自身を複製する. 今回のボットがダウンローダなどであった場合は，潜伏先フォルダ内にボットの本体が格納される. 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’ ）が付けられる場合が多い.
- 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，ボットは侵入挙動は行わず，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併せ持っており，実行環境に応じて自らの挙動を変化させている[3]. 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’ を起動した場合においても，実行環境を感染初期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ことを意味する.

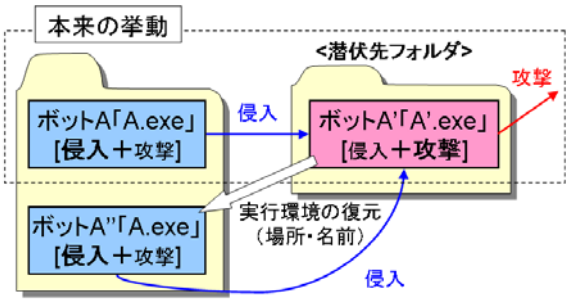

図 1：高機能ボットにおける侵入挙動の反復性

本稿では，上記の侵入挙動のうち，まずは実行場所の変化[3]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.

● 低機能ボット

低機能ボット B は，常に侵入・攻撃挙動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. すなわちボット B は，潜伏先フォルダへの侵入後も常に侵入挙動を行い続けるため，高機能ボットのときのように意図的に実行環境を操作せずとも侵入挙動の反復が観測できると予想される.

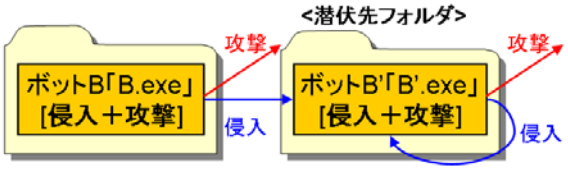

図 2：低機能ボットにおける侵入挙動の反復性

2.2 正規プログラムにおける挙動の反復性

インストーラやコンパイラなどは任意のフォルダに実行ファイルを生成するため，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（例えば，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.

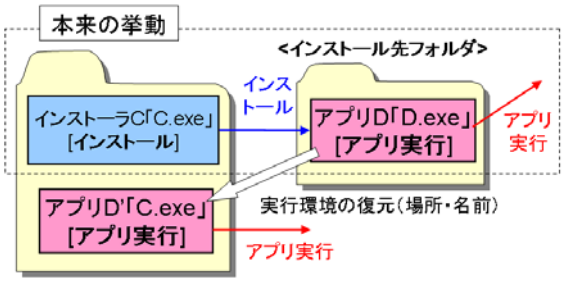

図 3:正規プログラムにおける侵入挙動の反復性

3 提案方式

本章では，侵入挙動の反復性を特徴的なビヘイビアとしてボットを検知する方式を説明する. 本方式の概要を図 4 に示す.

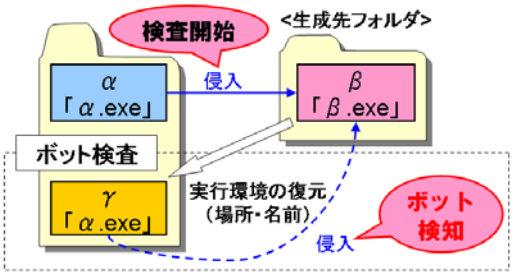

図 4:提案方式の概要

まず，実行中の全プロセスを監視し，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ボット検査を行う. この時，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. 次に，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. 2.1 節で示したように，本稿では，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. また，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. 最後に，γ を実行させた際に再び侵入挙動(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 及び β をボットとして検出する.

2 章の説明のうち，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. 一方，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. しかし本稿では，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.

なお，本方式を実現するためには，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，file access や crate file 等のイベントをリアルタイムで監視することによって可能となる.

4 検証

4.1 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に検知する必要はないため，ファイルアクセスに関する API をフックする代わり

に，プロセスのファイルアクセスを監視可能なモニタツールである ProcMon [4]を用いてファイル生成のイベントを抽出することとした．同様に，実行ファイルの自動実行リストへの登録に関しては，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 Autoruns[5]を用いて確認することとした．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．

4.2 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 2009[6]のマルウェア検体 10 個である．表 1 に本実験で用いたボットと実験結果を示す．表中の「タイプ」とは，2.1 節で説明した高機能ボット・低機能ボットの区別を示している．また，高機能ボットにおける「場所」と「名前」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

表 1:検知実験結果

ボット検体	タイプ			判定
	高機能		低機能	
	場所	名前		
検体 A	－	－	○	○(検知)
検体 B	×	○	－	○(検知)
検体 C	－	－	○	○(検知)
検体 D	－	－	○	○(検知)
検体 E	○	○	－	○(検知)
検体 F	○	○	－	○(検知)
検体 G	－	－	○	○(検知)
検体 H	－	－	○	△(一部検知)
検体 I	×	×	×	×
検体 J	×	×	×	×

表1に示すように，A～G の7体の検体においては，2.1 節にて説明したいずれかの挙動が観測され，本方式で検出することができた．検体 H においては，実行ファイルを生成する挙動は見られなかったが，実行のたびに自身を自動実行リストに登録する挙動が観測されたため，侵入挙動の反復性が一部観測されたものとし，「一部検知」としている．これは，侵入挙動の定義を見直すことによって検知できる可能性がある．また，検体 I，J においては，攻撃機能のみを備えるボット本体を生成するタイプの検体であり，検体の起動によって生成された実行ファイルが侵入機能を持っていなかったため侵入挙動の反復性が観測されなかった．

以上のように，侵入挙動と攻撃挙動を併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．

4.3 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を正規のプロセスとして採用した．表 2 に本実験で用いた正規プログラム(インストーラ)と実験結果を示す．また，正規プログラムのうち Rainlendar はカレンダー管理ツール[7]，ExtendQuickBar はクイックバー管理ツール[8]のフリーソフトである．実験結果を表 2 に示す．

表 2:誤検知実験結果

正規プログラム	判定
MS WORD2003	×
InternetExplorer	×
Adobe Reader	×
Skype	△(一部誤検知)
Rainlendar	△(一部誤検知)
ExtendQuickBar	△(一部誤検知)

MS WORD2003，InternetExplorer，Adobe Reader のインストーラについては，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. 3 章に記したように, 提案方式は, 実行ファイル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ることもなかった.

Skype, Rainlendar, ExtendQuickBar については, 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, ユーザが設定変更の操作を行った時点で実行ファイル(γ)が自分自身を自動実行リストに登録するという挙動が観測された(γが, さらに実行ファイルを生成することはなかった). これは, 表 1 の判定基準に従えば,「一部誤検知」となる. これらの正規プログラムのように, PC に常駐するタイプのアプリケーション(図 4 におけるβ)においては, 自分自身を自動実行リストへ登録するかどうかの選択肢が存在していることが多々ある. しかし, 正規プログラムにおいてはユーザの意思によって自動実行リストへの登録を選択するように作られていることが一般的であり, 起動のたびに毎回無条件に自分自身を自動実行リストに登録するような挙動を示すことは少ないと考えられる. よって, 条件を整理することによって, 検体 H は検知し, 正規プログラムは誤検知しない方式へと改良できる可能性があるのではないかと考えている.

4.4 考察

検証実験によって侵入機能と攻撃機能を併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が確認できた. しかし, 検体 H のように正規プログラム(インストーラ)に近い挙動を見せるものもあり, 侵入挙動およびその反復性の定義をさらに検討する必要があると考えられる. また, 検体 I, Jのようにそれぞれの機能を単体で有するタイプのボットについては本方式では検知することはできない. ボットはこのように多種多様なタイプに分かれるため, 一つの方式で全てのボットを検出することは困難であると考えられる. そのため, ボットのタイプごとに, 適した検知方式を導出する必要がある. 本稿では, そのうちの一タイプに対して, 有効な検知方式を提案できたと考えている.

5 まとめ

本稿では, ボットが侵入機能と攻撃機能を併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. 監視ツールを用いた基礎実験から, 本方式によるボット検出の有効性を確認することができた. 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検知精度を高めるとともに, 侵入機能と攻撃機能を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.

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] 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

[3] 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)

[4] Microsoft TechNet, "ProcMon", http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx

[5] Microsoft TechNet, "Autoruns", http://technet.microsoft.com/ja-jp/sysinternals/bb963902.aspx

[6] 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(2009.10)

[7] Rainy "Rainlender", http://www.rainlendar.net/cms/index.php

[8] mebiusbox software, "ExtendQuickBar", http://mebiusbox.crap.jp/software_extend_quick_bar.html