Vol.2009-CSEC-46 No.16
2009/7/2

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

酒井崇裕[†1] 竹森敬祐[†2] 安藤類央[†3] 西垣正勝[†4, †5]

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．そこで本研究では，ボットの多くがシステムフォルダ内への侵入と指令に従った攻撃という二つの挙動を併せ持つという特徴に注目した．ボットは自分の実行環境に応じて自らの振る舞い（侵入と攻撃）を変える．本稿では特に，ボット自身が潜伏先フォルダ内に存在するか否かによってボットの侵入／攻撃の挙動が変化することを利用し，実行場所による挙動変化を特徴的なビヘイビアとしてボット検出を行う方式を提案する．

A bot detection based on behavior changing according to execution environment

TAKAHIRO SAKAI[†1]　KEISUKE TAKEMORI[†2]
RUO ANDO[†3]　MASAKATSU NISHIGAKI[†4, †5]

Because of 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. In this paper we focus on the fact that almost all bots have both behaviors of intrusion and attack to achieve their aim, and creatively use both behaviors depends on their execution environment. Thus, we propose a new bot detection scheme which is based on bot's behavior changing according to execution environment. At the first step of the research, this paper discusses to detect bots by checking if their behavior is different when they are residing inside or outside of the system directory on the OS.

[†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，
Graduate school of Informatics, Shizuoka University, 3-5-1 Johoku, Naka, Hamamatsu, 432-8011 Japan
[†2] 株式会社 KDDI 研究所，〒356-8502 埼玉県ふじみ野市大原 2-1-15，
KDDI R&D Laboratories, Inc. 2-1-15 Ohara, Fujimino, Saitama, 356-8502 JAPAN
[†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，National Institute of Information and Communication Technology, Tracable Network Group　4-2-1 ,Nukui-Kitamachi, Koganei,Tokyo 184-8795 Japan

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，ボットと正規プロセス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，ボットが初めに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C&C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．この侵入と攻撃という 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，すべてのボットが上述の侵入／攻撃の両機能を単一の検体[*2] の中に持っており，状況に応じてその振る舞いを変化させるといっても過言ではないだろう．

ボットは，まず何らかの方法でターゲットとなるPCの任意のフォルダに入り込む．そして，任意のフォルダ内のボットが起動されることによって，PCはボットに感染する．この時点においては，侵入挙動によって自身の実行ファイルを潜伏先フォルダ内

[†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，
Graduate School of Science and Technology, Shizuoka University, 3-5-1 Johoku, Naka, Hamamatsu, 432-8011 Japan
[†5] 独立行政法人 科学技術振興機構，CREST，Japan Science Technology and Agency, CREST

[*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．

[*2] 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成る複数のボット郡が全体で一つのボットとして動くこともある．この場合は，「単一の検体」を「単一グループの検体群」と読み替えればよい．

に複製[*3]し，その後，潜伏先フォルダ内の実行ファイルが起動されて攻撃挙動を行うという段階的な挙動変化が観測されることが一般的である．また，その際にボットはPCのレジストリ等の書き換えも行う．これにより，それ以降は，感染PCがブートされる度に潜伏先フォルダ内のボットの複製が自動的に起動されるようになる．この時点では，通常，攻撃挙動のみが観測される．この事実から，ボットはレジストリの内容や実行ファイルの所在地などといった「自身の実行環境」から自分の状況を判断し，それに応じて挙動を変えているということが確認できる．

著者らは，このような実行環境による挙動変化が，ボットとして特殊なビヘイビアである可能性が高いと考えた．本稿では特に，ボット自身が潜伏先フォルダ内に存在するか否かによってボットの侵入／攻撃の挙動を変化させることを利用し，実行場所による挙動変化を特徴的なビヘイビアとしてボット検出を行う方式を提案する．

2. 実行場所による挙動変化

本章では，実行場所（実行プログラムが存在するフォルダ）を変化させた際の挙動の違いについてボットと正規プロセスの両方に対して確認する．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，ボットを検出し得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 実行場所によるボットの挙動

ボットが自身の実行ファイルの所在地によって侵入挙動と攻撃挙動を変化させているのならば，ボットを意図的に別の場所に移動して起動させた場合には図 1 のような挙動となると推測される．

まず，ボットを起動して初期感染させる．このとき，ボットは侵入挙動によって潜伏先フォルダに実行場所を移した後に攻撃挙動を行う．次に，潜伏先フォルダに複製されたボットの実行ファイルを起動させる．この場合は，ボットは侵入挙動は行わず，直接，攻撃挙動を開始する．更に，潜伏先フォルダにあるボットの実行ファイルを意図的に他の任意のフォルダへコピーし（図 1 の緑線），コピーした実行ファイルを起動する．別フォルダにコピーされたボットは，自身が潜伏先フォルダに存在していないことから再び侵入挙動を行う．このようにボットの場合は，実行場所の変化によって侵入挙動と攻撃挙動が切り替わると考えられる．

[*3] この時点で起動された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．

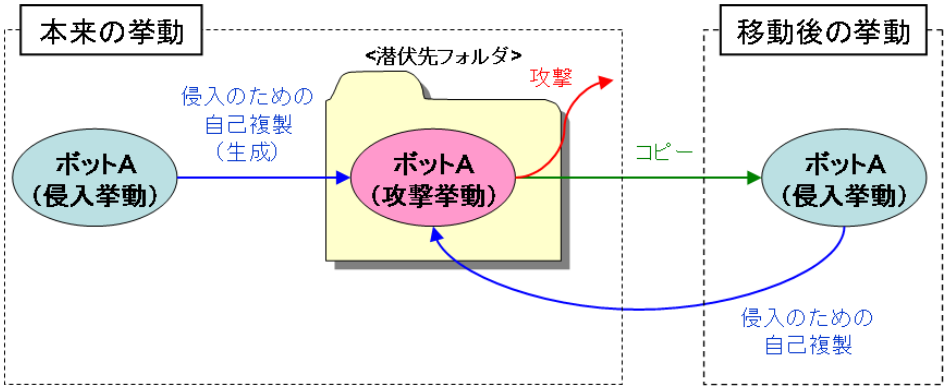

図 1　実行場所によるボットの挙動
Figure 1　Bot's behavior according to execution location

2.2 実行場所による正規プロセスの挙動

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．しかし，インストーラやコンパイラによって生成された実行ファイルを意図的に別の場所に移動させて起動した場合の挙動（図 2）は，ボットの挙動とは大きく異なると考えられる．

まず，インストーラやコンパイラなどによって生成された実行ファイルを実行すると，その実行ファイルの目的に応じた挙動が観測される（例えば，ワープロソフトであれば新規文書の作成画面が表示される）．次に，生成先フォルダ内の実行ファイルを意図的に別のフォルダへコピーし（図 2 の緑線），コピーした実行ファイルを起動する．一般的に言って，正規プロセスはボットと異なり，実行場所の変化を想定して作られていないため，実行ファイルを移動させてから起動したところで実行結果に違いは見られない．実行ファイルが参照するファイルのパス指定などの影響で実行場所が限定される可能性はあるが，その場合，エラーによって実行できないことはあっても，別の挙動に変化することはないと考えられる．

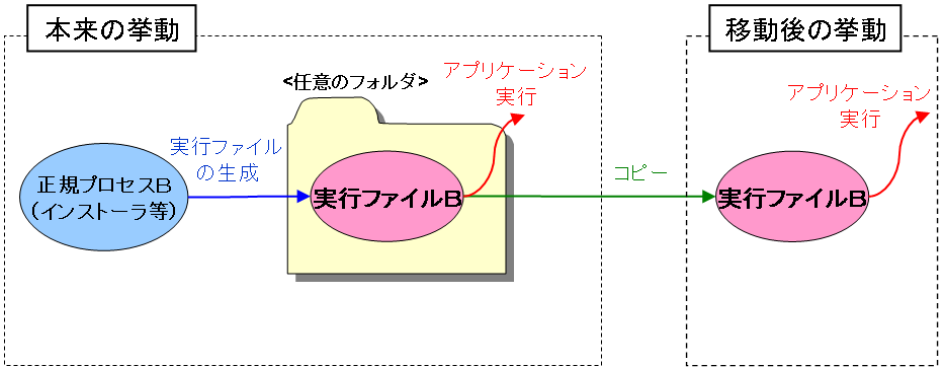

図 2　実行場所による正規プロセスの挙動
Figure 2 Legitimate program's behavior according to execution location

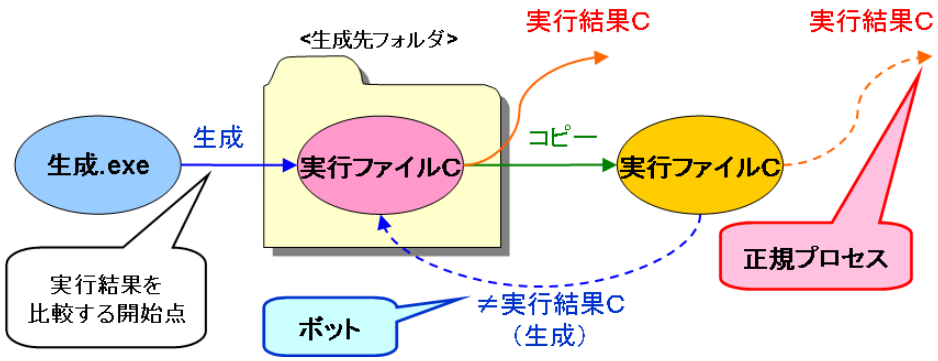

図 3　提案方式の概要
Figure 3 Overview of proposed method

以上のように，実行場所による挙動の変化には，ボットと正規プロセスの間に大きな違いがあると予想されるため，誤検知無く両者の切り分けができると期待される．

3. 提案方式

本章では，実行場所による挙動の変化の有無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 3 に示す．

まず，あるプロセスが実行ファイルを生成した瞬間を起点として，生成された実行ファイルに対して以下の挙動を観測する．

（1）生成された実行ファイルを生成先フォルダ内で実行させたときの挙動
（2）生成された実行ファイルを生成先フォルダ外へコピーしてから実行させた時の挙動

次に，この両者の挙動を比較する．比較結果が同じであった場合には，生成された実行ファイルは実行場所に依存することが無いため正規プロセスであると考えられる．比較結果が違う場合には，自らの実行場所を判断して挙動を変化させたと考えられるためボットとして検出する．なお，図中の「生成.exe」とは，実行ファイルを生成する任意のプロセスとする．

本方式を実現するためには，まず実行ファイルが生成された瞬間を検出する必要があるが，これは API をフックすることでリアルタイムに監視することが可能となる．また実行結果の比較には，各実行ファイルを実行させたときのファイルアクセスや通信をログとして記録する必要がある．これもまた，API フックにより実現可能である．侵入挙動時のファイル・レジストリの書き込みや攻撃挙動時の外部との通信などがボットにおける主な挙動となってくるため，これらの観点から実行ログを比較することが妥当だと考えられる．

なお，生成先フォルダの内外で実行ファイルを実行させる際に，ファイル等への書き込みに関する API を制御して無効化することで，PC をサンドボックス化して検証を行うことが可能である．ただし，本検知方式は，生成.exe によって生成先フォルダに実行ファイルが生成されていることが前提となっているため，生成.exe がボットであった場合，初期感染には成功する．

本方式は，すでに PC 内に存在している任意の実行ファイルに対して実行場所を変更してその挙動を比較すれば，実行ファイルの生成を起点とせずともボット検知が可能である．しかし，PC 内の実行ファイルは無数であり，どのファイルを検査すべきかの判断は困難を極める．そのため，検査対象の実行ファイルを特定するために，実行ファイルの生成を本方式の起点として捉えることは理に適っていると考える．

4. 検証実験

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．実行ログの比較に際しては比較情報が膨大となるため，今回は検査の簡略化のために，特にファイル生成のみに注目して実行ログを比較するものとする．検証に当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 [3]を用いる．このツールを用いて，検査の起点（プロセスが実行ファイルを生成したか）を見つけるとともに，その実行ファイルを当該フォルダ内外で実行させた場合に生成されるファイルの比較を行った．

具体的には，まず ProcMon により PC 内で発生したすべてのファイルアクセスを監視し，任意のプロセスが実行ファイルを生成しているかどうかチェックする．新たな実行ファイルαの生成が認められた時点で，その実行ファイルαを検査対象としてセットする．そして，実行ファイルαが生成されたフォルダ（生成先フォルダ）以外のフォルダを「コピー先フォルダ」として設定し，実行ファイルαをコピー先フォルダにコピーする．次に，実行ファイルαを生成先フォルダ内にて実行し，その結果を ProcMon により観測する．続いて，実行ファイルαをコピーした実行ファイルβをコピー先フォルダ内にて実行させ，その結果を ProcMon により観測する．両者の観測ログを比較し，ファイル生成に関する挙動が異なる場合にはボットの疑いありと判断する．ここで，ファイル生成の検出にあたっては CreateFile()の API を観測する．また，今回はコピー先フォルダを C ドライブ直下（C:¥）とする．なお，実行ファイルαとβは同一ファイルであるため，同時に実行させた場合にプロセスどうしが干渉しあう可能性がある(例えば,二重起動防止機能を有するソフトウエアなどがそれにあたる)．そのため，実行ファイルαとβは独立に実行させる．

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 PC は仮想マシンであり，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，著者らの研究室で独自に収集した検体 10 体である．

検知実験においては，ProcMonによりPC内で発生したすべてのファイルアクセスを監視している状態で，実験実施者が検体を起動させた．これによって，ボットは侵入挙動を開始し，潜伏先フォルダ（4.1 節の説明における「生成先フォルダ」に相当）に自分自身の実行ファイルの複製[*4] を生成する．これが起点となり，「ボットの複製（4.1 節の説明における「実行ファイルα」に相当)」が検査対象としてセットされ，4.1 節で示した手順でボット検知が実行されることになる．

検知実験の結果を表 1 に示す．

表 1　検知実験の結果
Table 1　Experimental results for bot detection

タイプ	ファイル生成に関する挙動			判定
	検体の実行により生成された実行ファイル	検査対象の生成先フォルダ内での実行により生成されたファイル	検査対象のコピー先フォルダ内での実行により生成されたファイル	
A (5/10 個)	C:¥WINDOWS¥System32¥spoolsvc.exe	なし	C:¥WINDOWS¥System32¥algs.exe C:¥mpaw.bat	○ (検知)
B (2/10 個)	C:¥WINDOWS¥System32¥firewall.exe	C:¥(省略~)¥Temp¥~19.tmp.exe	C:¥WINDOWS¥System32¥lssas.exe C:¥vxfewb.bat C:¥(省略~)¥Temp¥~1B.tmp.exe	○ (検知)
C (3/10 個)	C:¥WINDOWS¥System32¥urdvxc.exe	C:¥WINDOWS¥Help¥jbnshhqj.exe C¥Progmram Files¥NetMeeting¥rsewzjqn.exe C:¥WINDOWS¥Web¥wcxjhhj.exe : （以下，複数のファイルを生成）	C:¥WINDOWS¥System32¥urdvxc.exe	○ (検知)

10 体の検体はその挙動が大きく三つに分かれたため，それぞれをタイプ A～C とした．以下，タイプ別に説明を行う．

[*4] 検体がボットの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生成される．

- タイプ A

検体を実行したところ，C:¥WINDOWS¥System32 に新たな実行ファイル spoolsvc.exe が生成された．よって，これを検査対象とし，実行場所を変えて挙動を比較した．

生成先フォルダ内の実行ファイル（C:¥WINDOWS¥System32¥spoolsvc.exe）を実行したところ，ファイルは何も生成されなかった．一方，この実行ファイルをコピー先フォルダへコピーした上で実行（C:¥spoolsvc.exe）させた場合には，再び C:¥WINDOWS¥System32 へファイル algs.exe を生成するという挙動が見られた．ここで，spoolsvc.exe と algs.exe はファイル名が異なるだけで，同一ファイルであった（algs.exe に対して同様の実験を行っても，同じ結果が得られる）．また，C:¥mpaw.bat は実行ファイル C:¥spoolsvc.exe を削除するためのバッチファイルである．

このように，実行場所によって異なるファイル生成の挙動が観測できており，検体はボットとして検出できたといえる．

- タイプ B

検査対象の生成先フォルダでの実行（C:¥WINDOWS¥System32¥firewall.exe）およびコピー先フォルダでの実行（C:¥firewall.exe）の両方で，一時フォルダ（Temp フォルダ）に実行ファイルが生成されている．Windows では一般的に，一時フォルダ内のファイルに対しては OS から取得した一意なファイル名を付加する．このため，本方式では，一時フォルダ内のファイル名の違いは無視するものとする．すなわち，ここで観測された一時フォルダ内のファイルは，（ファイル名が異なってはいるが）同一のファイルであると見なされる．よって，タイプ B においては，一部，実行場所に依存しないファイル生成が観測されたことになる．

しかし，検査対象をコピー先フォルダにて実行した場合には，一時フォルダ以外に対するファイル生成も観測されている．これらのファイルの挙動は，タイプ A で観測された挙動と同じものであった．よって，挙動全体としては，実行場所によって異なるファイル生成が観測できており，検体はボットとして検出できたといえる．

なお，一時フォルダに生成された~19.tmp.exe，~1B.tmp.exe を新たな検査対象として同様の検査を行ったところ，実行場所に関係なくファイル生成は観測されなかった．すなわち，これらは侵入機能を持つ実行ファイルではないのだと考えられる．一時フォルダに生成された実行ファイルは，おそらく攻撃機能の一端を担うものであろう．

- タイプ C

検査対象をコピー先フォルダで実行した場合（C:¥ urdvxc.exe）は，再び C:¥WINDOWS¥System32 に戻る挙動が観測された．一方，検査対象を潜伏先フォルダで実行した場合（C:¥WINDOWS¥System32¥urdvxc.exe）は，既存の HTML ファイルと対になるように実行ファイルが複数生成された．当該 HTML ファイルは C:¥WINDOWS¥System32¥urdvxc.exe によって改変されており，ブラウザ等で読むことで再び urdvxc.exe が潜伏先フォルダにコピーされるようになっている．

以上から，このタイプもボットとして検出できたといえる．

以上のように，今回の検証実験では，すべての検体において，実行場所の変化により異なる挙動を行うことが確認された．本実験を通じ，本方式がボット検知に有効であることが示された．

4.3 誤検知実験

本方式によって正規のプロセス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セスと実験結果を示す．

誤検知実験においては，インストール後の正規アプリケーションの実行プログラムを直接，検査対象としてセットし，4.1 節で示した手順でボット検知を実行することとした．よって，インストール先フォルダが 4.1 節の説明における「生成先フォルダ」に相当する．また，検知実験の際と同様，一時フォルダ内に生成されたファイルに対してはファイル名の違いは無視するものとする．

表 2 より，MS WORD で誤検知が発生していることがわかる．MS WORD に関しては，Microsoft Office の仕様により，複数バージョンの Office を併用した場合は WORD の起動時に「インストールの準備をしています」というメッセージとともに Windows インストーラが起動してしまうことがある[4]．本実験では，WORD の実行ファイルをインストール先フォルダの外に出したことで，正規にインストールされた WORD とは別の WORD が実行されたのだと認識されてしまったのだと考えられる．このため，インストール先フォルダ外での実行の際には，インストール先フォルダ内での実行の場合には生成されないインストール情報ファイル群が生成されてしまった．ただし，このインストール情報ファイル群の中には実行ファイルは含まれていなかった．ボットの場合は，侵入の際に実行ファイルを潜伏先フォルダに生成するので，観測ログの比較対象を実行ファイルに限定することで WORD の誤検知については回避が可能であると考えられる．

なお，MS OUTLOOK は，インストール完了後の初めての起動の際に初期設定ファイルを生成する．このため，挙動の比較の際には，初期起動か否かの条件を揃えておく必要がある．例えば，OUTLOOK をインストール先フォルダ内で初期起動させた後に，続けてその実行ファイルをインストール先フォルダ外で実行させた場合には初期起動とはならず，両者の実行結果が異なってしまう．これを防ぐには，本方式を適用する前に，インストール先フォルダ内で一度 OUTLOOK を実行させるなどの対応が必

要である．この問題は，他のアプリケーションでも発生する可能性があるため，今後の検討対象とする．

表 2　誤検知実験の結果
Table 2　Experimental results for false detection

アプリケーション	インストール先フォルダ内での実行により生成されたファイル	インストール先フォルダ外での実行により生成されたファイル	判定
MS WORD 2003	C:¥(省略~)¥Temp¥~DF3D37.tmp	C:¥(省略~)¥Temp¥~DF90F3.tmp	× (誤検知)
	C:¥(省 略 ~)¥Application Data¥Microsoft¥Templetes¥~$Nromal.dot	C:¥(省 略 ~)¥Application Data¥Microsoft¥Templetes¥~$Nromal.dot	
		インストール情報	
MS EXCEL2003	なし	なし	○
MS OUTLOOK 2003	(初期実行時) 初期設定ファイル群	(初期実行時) 初期設定ファイル群	○
	C:¥(省略~)¥Temp¥Mso7B.tmp	C:¥(省略~)¥Temp¥Mso7C.tmp	
Internet Explorer6.0	なし	なし	○
Adobe Reader 9.0	C:¥(省 略 ~)¥Application Data¥Adobe¥Acrobat¥9.0¥SharedDataEvents-journal	C:¥(省 略 ~)¥Application Data¥Adobe¥Acrobat¥9.0¥SharedDataEvents-journal	○

5. 考察

検証実験によって本方式の有効性を示すことができた．しかし，MS WORD における誤検知や MS OUTLOOK における初期起動の考慮などについては検討をしていく必要がある．これは言い換えれば，前者が実行結果の比較に用いる挙動に関して，後者が実行環境に関しての検討であるといえる．

実行結果の比較に関しては，本稿ではファイル生成にしか注目していない．ボットや正規プロセスの実行結果としては，ファイル生成以外にもファイル・レジストリアクセスや通信など様々な挙動が出力される．その中から適切な比較対象を抽出することができれば，本稿で検知した検体以外のタイプのボットや WORD の誤検知に対する更なる改善が期待できる．

実行環境に関しては，プロセスの挙動が変化する要因を更に検討する必要がある．MS OUTLOOK では，初期起動において設定ファイルの生成を行っているため，「設定ファイルの有無」も実行環境の違いであると考えなければならない．このように，挙動を変化させる要因は実行場所以外にも考えられるため，挙動変化に影響する環境要因を整理するとともに，その影響を制御して本方式を使用する必要があると考える．

6. まとめ

本稿では，ボットの実行場所による侵入／攻撃の挙動変化に着目し，実行場所を意図的に変えて実行させたときの実行結果に違いを比較することでボット検出が可能であるか検討を行った．プロセスのファイルアクセスを観測する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，挙動変化の要因となる実行環境や実行結果を比較するにあたっての比較項目を抽出・整理することで，本方式の精度向上を図りたい．

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] 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] Microsoft TechNet，“ProcMon”，http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx
[4] Microsoft サポートオンライン，“複数のバージョンの Office を Office 2003 と併用することに関する情報“，http://support.microsoft.com/kb/828956/