# 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

酒井崇裕[*1] 長谷巧[*2] 竹森敬祐[*3] 西垣正勝[*4,*5]

[*1]静岡大学情報学部, 〒432-8011, 静岡県浜松市中区城北 3-5-1

[*2]静岡大学大学院情報学研究科 〒432-8011 静岡県浜松市中区城北 3-5-1

[*3]株式会社KDDI研究所, 〒356-8502, 埼玉県ふじみ野市大原 2-1-15

[*4]静岡大学創造科学技術大学院, 〒432-8011, 静岡県浜松市中区城北 3-5-1, [*5] JST, CREST

**概要** ボットは PC 内に潜伏するため, 感染後のボットを探し出すことは一般的に難しい. よって, ボットの感染の瞬間を捕らえることが賢明であると考えられる. PC への感染に注目した場合, ボットとワームの活動は近しいため, ワーム検知の手法をボット検知に流用することが可能であると期待できる. 著者らは既にワームの感染を「自己ファイル READ」の有無によって検知する方法を提案している. そこで本稿では, 自己ファイル READ を検出することによって, ボットの感染時の検知が可能かどうか検討する. また, ボットの PC への潜伏は, 感染時に自分自身を隠蔽する挙動として観測される. したがって, 本稿では「自己ファイル DELETE」についても, ボット感染時の特徴的な挙動の一つとして検討対象に加える.

# A feasibility study of bot detection based on capturing self-initiated READ/DELETE behavior

TAKAHIRO SAKAI[†1] TAKUMI NAGAYA[†2] KEISUKE TAKEMORI[†3] MASAKATSU NISHIGAKI[†4,†5]

[†1]Faculty of Informatics, Shizuoka University. 3-5-1 Johoku, Naka, Hamamatsu, 432-8011 JAPAN

[†2]Graduate School of Informatics, Shizuoka University

[†3] KDDI R&D Laboratories, Inc. 2-1-15 Ohara, Fujimino, Saitama, 356-8502 JAPAN

[†4]Graduate School of Science and Technology, Shizuoka University. [†5]JST, CREST

**Abstract** Bot detection after being infected is usually difficult, since bot hides itself on the infected computer. Thus, it is preferable to capture the moment of bot infection. With regard to the way of infecting computer, bot is similar to worm. In other words, it is expected that the existing worm detection schemes can be applied to bot detection. For worm detection, we have already proposed a scheme based on capturing "self-initiated READ" behavior, which could be applicable to a variety of worms including unknown worms. Therefore this paper investigates how effectively bot infection can be detected by checking its self-READ behavior. In addition, we know that many bot removes itself from computer to cover up its infection. So, this paper focuses also on using its "self-DELETE behavior" to capture bot infection.

## 1 はじめに

ボットは PC 内に潜伏するため, 感染先 PC のファイルの削除や強制シャットダウンなど, 表立った行動はとらない. 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, 正規の通信になりすます

ように，正規プロトコルを装うとともに通信量を制御している．このように，一度感染し潜伏したボットを検知することは一般的に困難である．よって，潜伏後のボットを発見するのではなく，PC への感染の瞬間を捕らえることが賢明であると考えられる．

PC への感染に注目した場合，ボットとワームの挙動は類似しており，ワーム検知の手法をボット検知に流用することが可能であると考えられる．著者らは既にワーム特有の感染挙動を「自己ファイル READ」として定義した検知手法を提案している[1]．これはワームの感染活動の中で自分自身の実行ファイルを複製するにあたって，メモリ上で活動しているワームのプロセスが，ファイルシステム上にある自分自身の実行ファイルを READ するという挙動に着目した方式である．本稿では，この「自己ファイル READ」を用いた検知手法がボットにも適用可能かについて検討する．

また，ボットの場合は，PC に潜むというその特徴から，感染時に自らを隠蔽するという活動がみられることが多い．そこで，メモリ上で活動しているボットのプロセスが，ファイルシステム上にある自分自身の実行ファイルを削除するという挙動についても，「自己ファイル DELETE」として検討対象とする．

## 2 ボットの感染時の特徴

ボットの感染時の特徴として「自己複製」と「自己隠蔽」がある．

「自己複製」とは，ワームの感染時の挙動と同様，PC 起動時にボットが自動実行されるように，ボットがシステムフォルダに自分自身の実行ファイル（以下，自己ファイル）の複製を置くという挙動である．文献[1]で示しているように，この挙動に際しては，メモリ上で活動しているボットのプロセスが自身の複製を作るに当たって，コピー元のファイル（ボットの自己ファイル）を読み出し，コピー先であるシステムフォルダに書き込むという動作が必要であるため「自己ファイル READ」が発生する．

「自己隠蔽」は，感染の痕跡を消すために，ボットが（システムフォルダに自己ファイルの複製を置いた後に，オリジナルの）自己ファイルを削除するという挙動である．この挙動に際しては，メモリ上で活動しているボットのプロセスが自己ファイルを削除する必要があるため，「自己ファイル DELETE」が発生する．

以上のように，ボットはその感染活動の過程に，自己ファイルに対して READ／DELETE するという挙動が見られる．よって，自己ファイル READ／DELETE を「ボットらしい挙動」として定義し，これを用いてボットを検出する方法を提案する．

## 3 検知方式

本検知方式では，自己ファイル READ／DELETE を監視することでボットの検知を行う．具体的には，OS のファイルシステムに対するアクセスをフックすることにより，エンドユーザの PC における全プロセスのファイルアクセスを常時監視し，自己ファイル READ／DELETE を行ったプロセスをリアルタイム検知する．これにより，ボットが自身をシステムフォルダへと感染させた瞬間もしくは隠蔽のために自身を削除した瞬間にこれを発見することが可能である．

通常，実行プログラムはその一部分が欠落するだけで正常に動作しなくなるため，ボットが感染するにあたっては基本的に自分自身をそっくりコピーすることになる．そこで READ の場合は，自己ファイルのヘッダ領域を除くプログラム（以下，プログラム本体）のすべてを READ したプロセスが検出された時点でアラートをあげる．DELETE の場合は，単純に自己ファイルを DELETE したプロセスが検出された時点でアラートを上げる．

また，ボットが自己ファイルを READ／DELETE する方法としては，次の方法も考えられる．

(1) 別のプロセスを起動し，自己ファイルの READ／DELETE を依頼する．
(2) 他の既存プロセスに寄生し，自己ファイルの READ／DELETE を実行する．

本稿では，これらを「間接型自己ファイル READ／DELETE」と呼び，これらについても検知対象とする．

なお本稿においては，ボット検知の可能性を示すことが目的であるため，現時点では，検知速度，

オーバヘッド等の性能面に関する評価は考慮しない.

## 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. 検知の可否における検証に当たっては実際にシステムを実装する必要はないため, ファイルアクセスのフックの代わりに, 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 [2]を用いて自己ファイル READ／DELETE の検出を行うことにより実験を行った.

具体的には, ProcMon により PC 内で発生したすべてのファイルアクセスを記録し, プロセスが自己ファイルに対して, プログラム本体のすべてを READ, または DELETE するかどうかをチェックする. すなわち, パス名 A のファイルを実行することによって生起されたプロセスが, パス名 A のファイルに対して, プログラム本体のすべてを READ または DELEET した場合に, ボットの疑いありと判断する.

また, ProcMon はファイルアクセスに加え, OS によるプロセスの生成／終了などについても監視することができる. よって, 間接型自己ファイル READ／DELETE についても, (1)「別プロセスを起動する」タイプに関しては, プロセスの動きを追いながら上記の検査を実施してやれば, ProcMon によって同様にチェックすることが可能である. すなわち, パス名 A のファイルが生起した別プロセス B が, パス名 A のファイルのプログラム本体をすべて READ, または DELETE した場合に, ボットの疑いありと判断する. また, プロセス B が更に別プロセスを生起する場合は連鎖的にチェックする. ここで, 別プロセスの生起は ProcMon で Process Create の発行を監視する.

間接型自己ファイル READ／DELETE における (2)「既存のプロセスに寄生する」タイプに関しては, ProcMon による観測が出来ないため, 今回は検証項目から除外した.

本実験は, 物理的に隔離されたネットワーク上で行った. 隔離されたネットワーク上の PC でボットを実行し, ProcMon でボットのファイルアクセスおよびプロセス生成を監視する. なお, 実験に使用した PC は仮想マシンであり, その際用いた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.

表 1: 検知実験の結果

| ボット名 | 自己ファイル READ | 自己ファイル DELETE |
|---|---|---|
| 検体 A | ○ | ○ (間接型) |
| 検体 B | ○ | ○ (間接型) |
| 検体 C | ○ | ○ (間接型) |
| 検体 D | ○ | ○ (間接型) |
| 検体 E | ○ | ○ (間接型) |
| 検体 F | ○ | ○ (間接型) |
| 検体 G | ○ | ○ (間接型) |
| 検体 H | ○ | ○ (間接型) |
| 検体 I (CCC2008 検体) | ○ | × |

### 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 2008 のマルウェア検体 (以降、CCC2008 検体) 1 個と著者らの研究室で独自に収集した検体 8 個の計 9 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. 自己ファイル READ／DELETE それぞれに対して,「○」が観測されたこと,「×」が観測されなかったことを表す. また, 間接型自己ファイル READ／DELETE が観測された場合には「(間接型)」と追記してある.

表1が示すように, 今回の検証実験ではすべての検体において, 自己ファイル READ が観測された. また, CCC2008 検体以外の検体においては,

| 親 PID | PID | プロセス名 | ｵﾍﾟﾚｰｼｮﾝ | パス | 結果 | 詳細 |
|---|---|---|---|---|---|---|
| 1452 | 1424 | 0689(省略).exe | CreateFile | C:¥WINDOWS¥system32¥Isass.exe | SUCCESS | ㈱省略㈲ |
| ㈱中略㈲ | | | | | | |
| 1452 | 1424 | 0689(省略).exe | ReadFile | C:¥0689(省略).exe | SUCCESS | Offset:0, Length:65,536 |
| 1452 | 1424 | 0689(省略).exe | WriteFile | C:¥WINDOWS¥system32¥ Isass.exe | SUCCESS | Offset:0, Length:65,536 |
| 1452 | 1424 | 0689(省略).exe | ReadFile | C:¥0689(省略).exe | SUCCESS | Offset:65,536, Length:65,536 |
| 1452 | 1424 | 0689(省略).exe | WriteFile | C:¥WINDOWS¥system32¥ Isass.exe | SUCCESS | Offset:65,536, Length:65,536 |
| 1452 | 1424 | 0689(省略).exe | ReadFile | C:¥0689(省略).exe | SUCCESS | Offset:131,072, Length:65,536 |
| 1452 | 1424 | 0689(省略).exe | WriteFile | C:¥WINDOWS¥system32¥ Isass.exe | SUCCESS | Offset:131,072, Length:65,536 |
| 1452 | 1424 | 0689(省略).exe | ReadFile | C:¥0689(省略).exe | SUCCESS | Offset:196,608, Length:65,536 |
| 1452 | 1424 | 0689(省略).exe | WriteFile | C:¥WINDOWS¥system32¥ Isass.exe | SUCCESS | Offset:196,608, Length:65,536 |
| 1452 | 1424 | 0689(省略).exe | ReadFile | C:¥0689(省略).exe | SUCCESS | Offset:262,144, Length:7,680 |
| 1452 | 1424 | 0689(省略).exe | WriteFile | C:¥WINDOWS¥system32¥ Isass.exe | SUCCESS | Offset:262,144, Length:7,680 |
| 1452 | 1424 | 0689(省略).exe | ReadFile | C:¥0689(省略).exe | END OF FILE | Offset:269,824, Length:65,536 |

図1 : 自己ファイル READ を行うボットの観測結果の一部

| 親PID | PID | プロセス名 | オペレーション | パス | 結果 | 詳細 |
|---|---|---|---|---|---|---|
| 1452 | 1424 | 0689(省略).exe | CreateFile | C:¥oeoeemct.bat | SUCCESS | (省略) |
| 1452 | 1424 | 0689(省略).exe | WriteFile | C:¥oeoeemct.bat | SUCCESS | Offset: 0, Length: 202 |
| ㈱中略㈲ | | | | | | |
| 1452 | 1424 | 0689(省略).exe | Process Create | C:¥WINDOWS¥ system32¥cmd.exe | SUCCESS | PID: 1404, Command line : cmd /c ""C:¥oeoeemct.bat" " |
| 1424 | 1404 | cmd.exe | Process Start | | SUCCESS | Parent PID: 1424 |
| ㈱中略㈲ | | | | | | |
| 1452 | 1424 | 0689(省略).exe | Process Exit | | SUCCESS | (省略) |
| ㈱中略㈲ | | | | | | |
| 1424 | 1404 | cmd.exe | SetDispositionI nformationFile | C:¥0689(省略).exe | SUCCESS | Delete: True |
| ㈱中略㈲ | | | | | | |
| 1424 | 1404 | cmd.exe | SetDisposition InformationFile | C:¥oeoeemct.bat | SUCCESS | Delete: True |

図2 : 自己ファイル DELETE を行うボットの観測結果の一部

間接的自己ファイル DELETE も確認できた. ProcMon のログ出力の中の自己ファイル READ／DELETE の観測結果の一部をそれぞれ図1, 図2に示す.

自己ファイル DELETE の場合はすべて間接型の挙動が観測されている. これは, Windows において実行中の実行ファイルを削除することはできないことに起因する. 自己ファイルを DELETE するためには, ボットは別のプロセスに自己ファイルの DELETE を依頼する必要がある. 今回の検体では, 自己ファイルの DELETE を実行するためのバッチファイルを生成し, このバッチファイルをコマンドプロンプトから呼び出すことで自己ファイルを DELETE していることが観測できた.

CCC2008 検体においては, 耐「ボット検知」機能を備えており, VMWare や FileMon [4]を検知した時点で PC への感染活動を強制終了していた. このため, 感染成功後に行う自己ファイル DELETE は検知する事ができなかった. しかしこの検体は, なぜか, VMWare の存在を検知する前に自己ファイルの READ を完了させていたために, 自己ファイル READ を観測することができた. VMWare や FileMon の存在の有無をチェックした上で感染活動を行うようなボットであった場合には, 自己ファイル READ は観測できなかったであろう. そのため, より高度な耐「ボット検知」機能を有した検体に対しては, 本方式による検知が困難になることが予測される. しかし, そのようなボットに対しては, 耐「ボット検知」機能を逆用[3]してボットの感染を抑制するという方法が採れると考えられる.

### 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. 表2に本実験で用いた正規プロセスと実験結果を示す. 自己ファイル READ／DELETE それぞれに対して, 「NG」が誤検知されたこと, 「OK」が誤検知されなかったことを表す. また, 間接型自己ファイル READ／DELETE に対する誤検知の場合には「(間接型)」と追記してある.

表2：誤検知実験の結果

| 正規プロセス | 自己ファイル READ | 自己ファイル DELETE |
|---|---|---|
| MS WORD | OK | OK |
| MS EXCEL | OK | OK |
| Internet Explorer | OK | OK |
| Adobe Reader | OK | OK |
| インストーラ (wireshark-setup -1.0.2.exe) | NG | OK |
| アンインストーラ (wireshark のアンインストーラ | NG | NG (間接型) |
| Windows Installer (apache_2.2.9-win32 -x86-no_ssl-r2.msi) | OK | OK |

MS WORD, MS EXCEL においては, プロセスを起動させた後, しばらくの間のファイルアクセスをモニタリングしたが, 自己ファイル READ／DELETE は観測されなかった. Internet Explorer, Adobe Reader においては, プロセスを起動させた直後に自己ファイルの一部を READ するイベントが観測された. しかし, 自己ファイルを READ する対象がファイルのプログラム本体の数%にしか満たないもの, また, ファイルのヘッダ領域の一部のみを READ しているもののみであった.

インストーラ (wireshark-setup-1.0.2.exe) においては, インストール実行中に自己ファイルをすべて READ するイベントが観測された. また, 同アプリケーション (wireshark) のアンインストールを実行する際にも自己ファイルをすべて READ するイベントが観測された. wireshark のアンインストールにおいては, 間接型自己ファイル DELETE も観測された. アンインストールに際しては, 自己ファイル (アンインストーラ) を含め, アプリケーションを構成するファイルをすべて削除する必要があるため, 自分自身 (アンインストーラ) の複製をテンポラリフォルダに作成した上で, その複製が起動され, ファイルの削除を実行していた. このため, 複製の作成時に自己ファイル READ が, ファイル削除の際に間接型自己ファイル DELETE が観測された.

一方, インストーラであっても, Windows Installer においては, Windows が提供する msiexec.exe がインストール, アンインストールを実行するため, 自己ファイル (msiexec.exe) の READ および DELETE は観測されなかった.

本実験を通じ, 正規プロセスにおいては Windows Installer によらないインストーラおよびアンインストーラについて誤検知が発生することが確認された.

## 5 インストーラとの切り分け

4 章の実験より, 本方式は高い確率でボットを検知することが可能ではあるが, 同時にインストーラおよびアンインストーラ (以下, まとめてインストーラとのみ表記する) の誤検知が問題となることが示された. このため, 本方式においては, ボットとインストーラの切り分けが必須となる.

その方式の1つとして「ユーザのレスポンスによ

る判定」が考えられる．一般にアプリケーションのインストールはユーザの意思によって実行されるものである．たとえ悪意のない正規アプリケーションであっても，ユーザの意図しないインストールは望ましくない．よって，インストーラの実行時にユーザからのレスポンスを義務付ける．このようにすることで，自己ファイル READ／DELETE が検出された時点において，ユーザにアラートを表示し，ユーザからのレスポンスが有るか無いかによって，ボット感染かインストールかを切り分けることができると期待される．

現在，コンピュータと人間を区別する技術するとしては CAPTCHA [5]が一般的である．よって，正規のインストーラにおいては CAPTCHA の導入を義務付け，CAPTCHA によってユーザの存在を確認した上でインストール作業が実行されるようにするなどの方法が実用的であると考えられる．

## 6　まとめ

本稿では，ボットの感染時の挙動がワームに類似していることに着目し，著者らが以前に提案した自己ファイル READ によるワーム検知方式をボット検知に適用できるかについて検討を行った．また，ボット特有の隠蔽活動を考慮し，自己ファイル DELETE によるボット検知についても検討を行った．プロセスのファイルアクセスを観測するモニタツールを用いた基礎実験から，本方式によるボット検知の可能性を確認することが出来た．

## 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] 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 年 9 月．

[2] Microsoft TechNet，“ProcMon” http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx（2008 年 9 月確認)

[3] 松木隆宏村上純一，“悪性プログラムの耐解析機能を逆用した活動抑制手法の提案” CSS2006 論文集，pp.299-304(2006.10)

[4] Microsoft TechNet, "FileMon", http://technet.microsoft.com/en-us/sysinternals/bb896642.aspx（2008 年 9 月確認)

[5] Carnegie Mellon University, "The Official CAPTCHA Site", http://www.captcha.net/（2008 年 9 月確認)