クライアント側PCでのサーバ的振舞によるボット検出法

湯浅 紘一† 小林 和朝‡ 高田 寛之‡

† 長崎大学大学院生産科学研究科
852-8521 長崎市文教町 1-14

‡ 長崎大学工学部
852-8521 長崎市文教町 1-14

{kobayashi,htakada}@cis.nagasaki-u.ac.jp

あらまし 近年ボットと呼ばれるウイルスによる脅威が大きな問題となっている. ボットは感染した後, ボットネットを介した外部からの指令を受け, コンピュータ内の情報を盗む攻撃や外部のコンピュータへの攻撃を行う. 本稿ではクライアント側 PC 上でボットが外部への攻撃活動を行う場合のプロセス形態が, 通常のアプリケーションのそれと異なり, サーバとして動作するアプリケーションと似ていることに注目し, その特徴をビヘイビアとするビヘイビア法の検出法を提案する.

A bot detection based on server-like behavior of the client-side PCs

Kouichi Yuasa† Kazutomo Kobayashi‡ Hiroyuki Takada‡

†Graduate School of Science and Technology, Nagasaki University,
1-14 Bunkyo-Cho Nagasaki-Shi Nagasaki 852-8521 Japan

‡Faculty of Engineering, Nagasaki University,
1-14 Bunkyo-Cho Nagasaki-Shi Nagasaki 852-8521 Japan

{kobayashi,htakada}@cis.nagasaki-u.ac.jp

Abstract In recent years, a threat caused by computer virus called a bot is a serious problem. After a bot infects a computer, it receives some commands through a botnet and does some attacks such as strealing information in the computer or causing other hosts damages. In this paper, we pay attention to the point that when the bot attacks other hosts a form of its processes differs from that of client applications and is similar to that of server applications. Then we propose a bot detecting method based on the server-like behavior.

1 はじめに

近年, ボットと呼ばれるウイルスがインターネット上に広まっており, さまざまなサイバー犯罪の温床となっている. ボットはディスプレイに目立つような表示をしたり, ワームのように大量に通信を行ったりして自らの存在をひけらかすようなウイルスと異なり, 金銭的な利益をあげることを目的としているためパソコンのユーザからその存在を気づれにくくする機能を多数持っており, 検出が困難となっている. そのためボットの検出についてはパターンマッチング法よりもヒューリスティック法, ビヘイビア法などによる方法が効果的である [1]. ビヘイビア法は実際に対象プログラムを動作させてみて, そのときの挙動をウイルスの特徴的なビヘイビアと比較することで検出を

行う方法である. したがってビヘイビア法ではボットの特徴的なビヘイビアを見付けることが重要となる.

ボットの特徴として, ボットネットと呼ばれるネットワークを構成することが挙げられる. ボットは対象 PC に感染し発症した後, ボットネットを介して指示を受け活動を行う. この活動のうち, 外部との通信を行うものは感染したパソコンの情報を盗むものや外部への攻撃を行うものなどである。著者らはクライアントして利用されている PC 上で, 外部へ攻撃活動を行っているボットのプロセス形態がサーバとして動作しているアプリケーションのそれに似ている点に着目した. この特徴をビヘイビアとしたビヘイビア法の検出方式の有効性を確認するための検証実験を行った.

2　外部との通信とプロセス形態

本章では Linux プラットフォーム上で外部との通信を行うアプリケーションに関して, 通常のものとボットとの違いを述べる. 前提として, パソコン (以下,PC と呼ぶ.) の用途は一般のユーザが PC を使うのと同じと仮定している. つまりサーバなどとしては動作していない. 外部との通信を行っているアプリケーションについて考える. アプリケーションを作成する場合, その処理の流れが１つになることは少なく, 多くの場合マルチスレッドプログラミングが行われる. Linux 上でのマルチスレッドの実現方法としては子プロセスによるものやスレッドによるものなどがある. 子プロセスを使う場合, そのアプリケーションに関するプロセスの木構造ができる. スレッドを使う場合, 軽量プロセス[1]を使って実装が行われる [?]. 軽量プロセスはプロセスと同じ様に振る舞うので, ユーザからはアプリケーションが１つ以上のプロセスの集まりとして見える. これらのスレッドにも親子の関係があるので, 木構造ができる. このアプリケーションのプロセス集合から成る木構造は複雑な構成となることが考えられるが, プロセスの親と子それぞれが外部と通信を行っているがどうかの関係にのみ注目すると, 適当な位置でノードを切り離したり部分木をまとめることにより, 最大でも高さが２の木を考えるだけでよい. 以下ではこの木構造を使って Linux 上で外部との通信をするクライアント側の通常のアプリケーションとボットのプロセス形態の違いについて述べる. 特に断らない限り, アプリケーションの持つスレッドのことをプロセスと呼ぶ.

2.1　通常の外部と通信を行うアプリケーション

調べ物をしたり, 作業をしたりするのに PC を利用するとき, 外部との通信を行うアプリケーションはサービスを受ける側, つまりクライアントとして動作している. これらのアプリケーションではその木構造のプロセスの中のある１つのプロセスが外部との通信を行う. アプリケーションが１つのプロセスから構成させる場合, そのプロセスが通信をすることになる. 一方複数のプロセスからなる場合, どれか１つのプロセスがすべての通信を請け負うことになる. クライアントとして動作するアプリケーションはサービスを受けるために通信するだけでよく, １つのプロセスのみが外部との通信を行うことになる.

2.2　ボットのプロセス

ボットのプロセスが外部から指示を受けて, 攻撃活動を行うときのプロセスの親子の関係は図 1 のようになる. 図中で灰色のものが外部と通信をしていることを表している.

[1]親プロセスとメモリ空間やファイルなどを共有したプロセス. 親プロセスと共有しているのでコンテキスト切り替えなどが通常のプロセスの切り替えよりも早い.

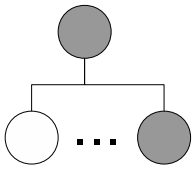

図 1: ボットのプロセス

攻撃活動を行っているときのボットのプロセスは図 1 のように親も子も外部と通信を行っている. 親はボットネットを介した指示を受けるための通信を行っており, 子は感染活動や DoS, ポートスキャンをするために通信を行っている. つまり親は要求を受けるために待機しており, 子は要求されたことを実行するために動作する. この関係は一般のサーバにおけるプロセスの関係と同じである. サービス要求を受けるために親プロセスが特定のポートで待機しておき, 実際のサービスを提供するのは子プロセスに任せる. 以上よりボットはサーバアプリケーションのとるプロセス形態と似た形態をとる.

3 特徴的ビヘイビアを用いた検出方式

本章ではクライアント側 PC 上でのボットのサーバ的振舞を利用したボット検出について述べる.

既述の通り, クライアント PC の通常のアプリケーションとボットとの間にはプロセス形態の違いが存在する. この特徴をビヘイビアとし, 次のようしてボットを検出する.

1. プロセスの監視
   ネットワークトラフィック解析プログラム [**?**] を使って外部との通信を行っているプロセスを監視する. プログラムの基本的なプロセス特定の流れを次に示す.
   (a) パケットキャプチャライブラリを使ってキャプチャしたパケットの IP アドレスとポート番号の情報を得る. 送信元が偽造されたパケットに関してはそのパケットが存在する時点で PC 内のボットの存在を検出できたとし, 次の段階には進まない.
   (b) 現在セッションを確立している通信の情報を保持しているシステムファイル[2]と先に得た IP アドレスとポート番号を比較し, そのソケットの inode 番号[3]を得る.
   (c) プロセスの情報を持つシステムディレクトリ[4]を探索し, 先に得たものと同じ inode 番号のソケットを持つプロセスを調べる.
2. 親プロセスの調査
   外部と通信をしていることを検出した全てのプロセスについてそのプロセスに関する情報の書かれたシステムファイル[5]を参照し, その親プロセスが他の通信をしているプロセスと同じかどうかを調べる. この調査はプロセスツリーの根にたどり着くまで行われる.
3. ボットのプロセスの検出
   先の調査の結果, 親の中に外部との通信を行っているプロセスがあればボット, そうでなければ通常のアプリケーションと判断する.

[2]TCP と UDP のソケットテーブルのダンプファイル (/proc/net/{tcp,udp}) のことである.
[3]ファイルを取り扱うのにファイルシステムが必要な情報を保持する構造体の識別番号.
[4]/proc ディレクトリ以下の数字からなるディレクトリのこと. この数字はプロセス ID(PID) を表す.
[5]/proc/[プロセス ID]/stat のファイルのことである.

4　実験

本章では提案したビヘイビアの有効性を検証するための実験について述べる.

4.1　検証実験

実験は次の 2 つの実験からなる. なお実験で使用した PC の OS は 64 ビット版 Fedora Core 10, カーネルは 2.6.27 である.

1. 通常のアプリケーションのプロセス形態を調べる実験.
   外部のホストとアクセスするアプリケーションを使い, そのときのプロセスの形態を調べる. この実験で使用したアプリケーションは以下の通りである.

   - Web ブラウザ (firefox バージョン 3.0.13)
   - メールソフト (evolution バージョン 2.0.0.23)
   - インターネットラジオ (beep media player バージョン 0.9.7.1)
   - リモートデスクトップ (vino バージョン 2.24)
   - P2P アプリケーション (ktorrnet バージョン 3.2.2)
   - 更新プログラム (yum バージョン 3.2.23)

2. ボットのプロセス形態を調べる実験.
   インターネット上で入手した kaiten というボットのソースコードをコンパイルした後,VMWare を使って構成した仮想ネットワーク上で動作させ, 攻撃の指示を出す. この実験でボットに行わせた攻撃は次の通りである.

   - UDP Flood
     サイズの大きな UDP パケットを大量に送りつける.
   - ポートスキャン
     任意のポートに接続を試み, 接続できるかどうかでポートの開放を判断する.

4.2　結果

実験の結果を表 1 に示す.

- Web ブラウザ, メールソフト, インターネットラジオ
  マルチスレッドとして実装してあった. 外部との通信をするのは 1 つのプロセスだけであった.
- リモートデスクトップ
  単独のプロセスで実装してあった. 制御情報や表示するための情報などの送受信を行っていた.
- P2P アプリケーション
  マルチスレッドとして実装してあった. 外部との通信をするのは 1 つのプロセスだけであった. 数 Mbps の速度でファイルをダウンロードするのでパケットの取りこぼしが見られた.

表 1: 実験の結果

	親子ともに外部と通信しているか
Web ブラウザ	していない
メールソフト	していない
インターネットラジオ	していない
リモートデスクトップ	していない
P2P アプリケーション	していない
更新プログラム	していない
ボット (UDPFlood)	している
ボット (ポートスキャン)	している

- 更新プログラム
  単独のプロセスで実装してあった.

- ボット
  全ての攻撃の実験において親プロセスは外部と通信を行っていた. UDP Flood, ポートスキャンともに子プロセスとして外部に攻撃を行っていることを検出できた.

5 考察

実験の結果, 本稿で提案した通りのプロセス形態でボットが活動していることがわかった. ボットのプログラムは 2.2 節で述べた通りの挙動をしたため検出ができた. Web ブラウザやメールソフト, 更新プログラム, インターネットラジオはクライアントとして動作するアプリケーションなので 2.1 節にて述べた通りの挙動をとった. 一方リモートデスクトップや P2P アプリケーションなどはサーバとして動作するアプリケーションであるにも関わらず 2.2 節で述べたものとは異なる挙動をとった. 結局, ボット以外のすべてのアプリケーションにおいてそれを構成するプロセスのうちの 1 つが外部との通信を担当するという結果となった. 今回の実験の結果がこのようになった原因は Linux におけるスレッドの実現方法にあった.

既述の通り Linux ではスレッドを実現するために軽量プロセスを使う. 軽量プロセスはその親プロセスとアドレス空間や物理メモリページ, オープンされたファイルなどを共有できる. Linux においてはソケットもファイルとして扱われるため, すべてのスレッドで共通のソケットが使われることとなる. すべてのスレッドが共有する部分を持っているのは 1 つ目のスレッド[6]である. 第 3 章で述べた通りこのプログラムではある通信をするのに使っているソケットを持つプロセスを特定しているだけなので, どのスレッドが外部との通信を行っていても, 1 つ目のスレッドが通信をしているという結果になってしまったのである. 一方ボットについてはスレッドではなく子プロセスを作って攻撃を行わせていたため, そのプロセスが外部と通信をしているとして検出できたのである.

今後はシステムコールをフックすることによって, どのスレッドが外部と通信をしているのかを明らかにしたいと考えている. また今回の提案方式がスレッド間の親子関係を考えた場合にも有

[6] プログラムの中で一番初めに作られたスレッドである. つまりそのアプリケーションのすべてのスレッドの親に当たるスレッドである.

効であるのかを検討したいと考えている.

6 まとめ

本稿ではサーバとして利用することを考えないクライアント PC でのボットのプロセスのサーバ的振舞に着目したボット検出法を提案した. ボットに関しては提案通りの挙動をとったため検出ができた. 通常の外部と通信をするアプリケーションについては, どのスレッドが通信をしているのかが特定できていないとわかったため提案通りの挙動をとるのかの確認ができなかった. 今後はシステムコールをフックする方法でのプロセスの監視, スレッド間の親子関係の特定が課題である.

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] DANIEL P.BOVET,MARCO CESATI, 高橋 浩和, 早川 仁, 岡島 順治郎,"詳細 Linux カーネル", オライリージャパン,2002