社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－122（57）
2005－CSEC－28（57）
2005／3／23

感染プロセスに着目したワーム検知方式の提案

前田 秀介　馬場 達也　大谷 尚通　角 将高　稲田 勉
(株) NTT データ 技術開発本部
〒104-0033 東京都中央区茅場町 1-21-2 茅場町タワー
e-mail : {maedasu, babatt, ootanihs, kadom, inadatt}@nttdata.co.jp

概要：近年, Blaster や Sasser などの“ワーム”による被害が深刻化している．ワームを検知する代表的な手法としてシグネチャマッチング方式があるが，この方式では，既知のワームしか検知できないという問題がある.このため，“ワームらしさ”を検知することで未知のものも含めて検知する方式がいくつか提案されている．しかし，これらの方式は誤検知（false positive）が多く，必要な通信の遮断や，不要な警告の過多による対処遅れの原因になるなどの問題がある．本稿ではワームが感染の際に必要とするプロセス（挙動とその順序）に着目することにより，誤検知を抑えつつ，効果的にワームを検知する方式を提案する．
キーワード：ワーム，侵入検知システム(IDS)，侵入防御システム(IPS)，振る舞い検知，ビヘイビアブロッキング

A Proposal of Worm Detection Method Following the Infection Process

Shusuke MAEDA, Tatsuya BABA, Hisamichi OHTANI, Masataka KADO, Tsutomu INADA
Research and Development Headquarters,
NTT DATA Corporation

Abstract: The network incidents caused by Internet worms are increasing every year. The signature matching method for worm detection is effective only for known-worms, and is not effective for their variants and unknown-worms. Several methods for unknown-worm detection have been proposed. These methods detect worm’s typical behaviors. However, such methods generate many false positives. In this paper, we propose an improved worm detection method following the infection process that is necessary for Internet worms to spread.
Keyword: Internet Worm, Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Behavior Detection, Behavior Blocking.

1. はじめに

近年，ネットワークを通じて感染を拡大する自己増殖プログラム“ワーム”が問題となっている[1]. 特に Blaster や Sasser のように OS の脆弱性を悪用して増殖するネットワーク型ワーム(以下，OS 脆弱性悪用型ワーム)は，メールによって感染するものと異なり，端末が起動していれば人手を介さずに感染することが可能であり，非常に高速に感染が拡大してしまうという問題がある.

脆弱性は，その脆弱性をふさぐためのパッチを適用することによって感染を防ぐことができるが，企業などにおいては，業務システムへの影響が懸念されるためパッチを当てられない場合もある．また，パッチがシステムへ与える影響を調べるための検証期間中にワームに感染してしまう事例もある．実際，2003 年に Blaster が発生して問題になった時期には，パッチを適用する前や，ウィルス対策ソフトのウィルス定義ファイルをアップデートする前に Blaster に攻撃され，端末起動直後に感染してしまうといったケースが多く報告された．脆弱性の公表から実際にワームが出現するまでの期間は年々短くなってきており，脆弱性に対するパッチがリリースされる以前に発生する“ゼロデイ(Zero Day)アタック”も脅威となっている.

これらの問題への対策として，ワームの通信の特徴を検知することで，次々と出現する新種や亜種を防ぐ方法が考案されている．しかし誤検知が多く，遮断すべきでない通信まで遮断してしまったり，不要な警告が多すぎるために問題への対処が遅れてしまったりするといった問題がある．実用レベルで企業ニーズを満たすような方法は未だ現れていないのが現状である.

本稿では，ワームが感染するために必要とするプロセス(挙動とその順序)に着目してトラフィックを観測し，ネットワーク内のワーム感染端末を誤検知なく検知するアルゴリズムを提案し，検証実験によってその有効性を示す.

2. 従来のワーム対策とその問題点

今日まで，様々なネットワークベースのワーム検知方式が提案されている[2]. もっとも一般的なものに，既知のワーム情報と一致する通信を検知するシグネチャマッチング方式がある[3]. この方式は既知のワームの通信に関する情報をシグネチャとして保存しておき，該当する通信を検知するものである．しかし，この方式は既知のワームについては検知できるが，全くの新種はもちろん，通信を行うポートが異なるだけの亜種であっても対応できないといった問題がある.

そこで，シグネチャだけではなく，ワーム特有の挙動に着目した検知方式が注目されており，統計的アノマリ検知方式やプロトコルアノマリ検知方式などが提案されている.

統計的アノマリ検知方式はあらかじめ通常の通信を学習しておき，学習内容から逸脱する通信を検知する方式である[4]. この方式ならばシグネチャによらずにワームを検知できる．しかし，コンピュータの通信は複雑であり，通常状態をプロファイル化することは難しい．さらに，学習の際に異常なトラフィックが流れていた場合は，検知の精度が著しく低下する可能性がある．また，学習時には発生していなかった正常トラフィックを異常なものとして誤検知してしまう可能性もある.

プロトコルアノマリ検知方式はプロトコルの仕様から逸脱する通信を検知する方式である[5]. この方式はプロトコル違反を利用した攻撃を検知することができる．しかし，ベンダ独自のプロトコル拡張は珍しいことではなく，これをプロトコル違反とみなして誤検知が発生することが問題となっている．さらに，脆弱性の多くはプロトコルの仕様ではなくソフトウェアの仕様・設計ミスに因るものであり，プロトコル違反を検知するだけでは，ワームがよく利用するバッファオーバーフローなどの攻撃の検知は難しい.

以上のように，ワームの挙動に着目した検知方式は様々考案されているが，誤検知などの問題は未だ解消されていない．これは検知しようとしているワームの個々の挙動が通常の通信でも発生しうるものであることが原因だと考えられる．例えば，P2P ソフトウェアのセッションの張り方をポートスキャンとみなすこともできるし，FTP が制御セッションとは別にデータセッションを張りなおしてファイルの転送を行うことを，バックドアアクセスとみなすこともできる.ワーム感染を誤検知なく確実に検知するには，単に個々の挙動

に着目するだけでは不十分であると考えられる．

3. 提案手法

2 章で，従来のワームの特徴に着目した検知方式は単にワームの個々の挙動に焦点を当てているために誤検知が多くなることを示した．そこで，本章では個々の挙動に着目するだけでなく，さらにその順序を含んだプロセスという単位でワームの特徴を捉えることにより，誤検知を防ぎ，かつ確実にワームを検知する方式を提案する．

3.1. ワームの感染プロセス

OS 脆弱性悪用型ワームが他の端末に感染するときの挙動を以下に示す．

ワームが他の端末に感染するためには，まず感染対象となる端末を探さなくてはならない．そこでワームはポートスキャンを行い，感染対象となる脆弱性を持った端末を探す．

《N1》 ワーム感染端末 X はランダムに選んだ複数の IP アドレスの端末 Y_i (i=1,2,…,n_s) に対して，攻撃対象となるポート vp へのアクセスを試みる(図 1).

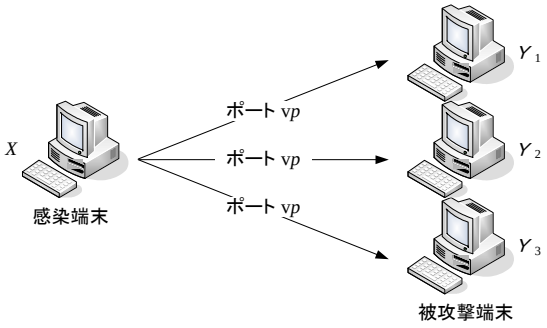

図 1：ポートスキャンによる感染先の探索

《N2》 ポートスキャンでアクセスが成功した端末 Y_j ($1 \leqq j \leqq n_s$)のポート vp に攻撃コードを送信し，Y_j の任意のポート ap でバックドアを開く．バックドアを開くことで端末 X は Y_j のポート ap に対して任意の命令を送信し，実行できるようになる．

通常，Windows のクライアント端末はファイルを受信するためのサーバ機能を有していない．そのため，端末 X から Y_j にワームプログラム $Prog$ を転送するためには，Y_j に X から $Prog$ をダウンロードさせる必要がある．そこでワームはあらかじめ端末 X のポート bp でファイル転送サーバ(FTP, TFTP など)を起動しておき，Y_j にダウンロードをさせる．

《N3》 端末 Y_j のポート ap に「X からワームプログラム $Prog$ をダウンロードする」という命令を送る．

《N4》 端末 Y_j が X のバックドアポート bp へアクセスし，ファイルダウンロード用のセッションを張る(図 2).

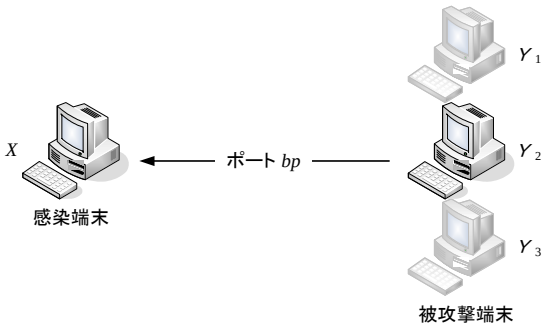

図 2：バックドアアクセスによるワームプログラム転送経路の確保

《N5》 端末 X から Y_j へワームプログラム $Prog$ が転送される(図 3).

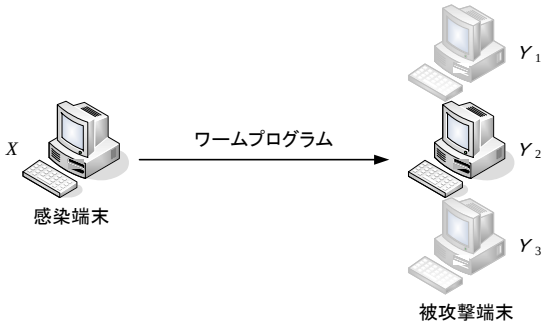

図 3：ワームプログラムの転送

最後に，端末 Y_j のポート ap に「ワームプログラム $Prog$ を実行する」という命令を送り，ワームプログラムを Y_j で実行することで感染が完了する．

《N1》のスキャンで感染対象を見つけなければ《N2》の攻撃コードを送ることはできない．また，《N2》でバックドアを開いておかなければ《N3》でダウンロード命令を送信しても受理されず，《N4》のダウンロードセッションが張られることもない．《N4》がなけ

れば，いきなり《N5》のようにファイルを送りつけても受理されることはない．以上のように，これらの挙動の 1 つが欠けても感染は成立しないことが分かる．この一連の挙動とその順序をワームの感染プロセスと呼ぶことにする．

例として実際に調査した Blaster.C と Sasser.C の感染プロセスにおける通信とプログラム実行の様子を図 4，図 5 に示す．これらの図は感染プロセスで用いられたセッションを上から下に時系列に並べたものである．図中のスクリプトとはバッチファイルであり，ファイル転送クライアントの起動や，ダウンロードしたファイルの実行などの命令が記述されている．

Blaster.C と Sasser.C の違いは，悪用する脆弱性が異なることと，ファイル転送に用いるプロトコルが FTP か TFTP かの違いである．また，Sasser.C は脆弱性攻撃を複数に分けて行っており，感染先の OS のバージョン確認なども行っている．しかし，本質的には上記の《N1》～《N5》に沿った動きをしている．他の亜種なども，使用するポートやスクリプトは違うものの，概ね同様の動作を行う．

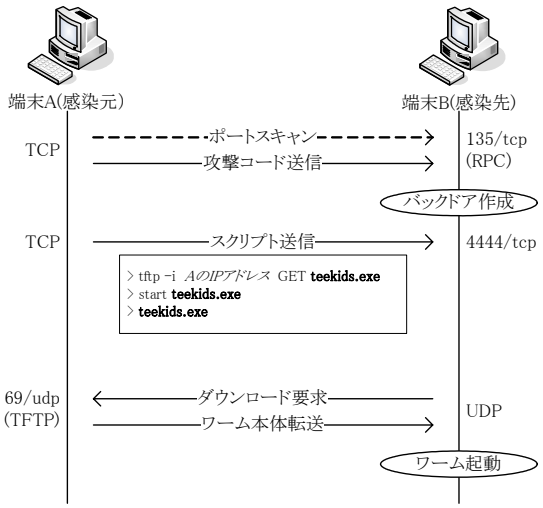

図 4 : Blaster.C の感染プロセス

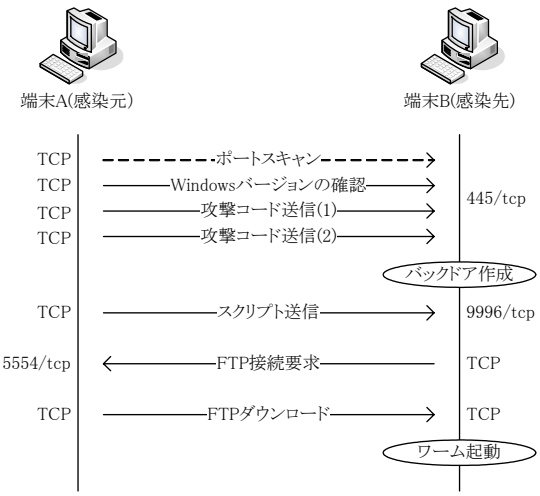

図 5 : Sasser.C の感染プロセス

3.2. 提案アルゴリズム

3.1 章で示したワームの感染プロセスをもとにワーム検知アルゴリズムを提案する．ワーム感染プロセスの中で送信される命令スクリプトは，サイズも小さく，その内容が不正アクセスに利用される命令かどうかを判断することは難しい．また，攻撃コードは対象となるプロトコルやソフトウェアの仕様を知っていなければ判別できない．そこでそれ以外の《N1》，《N4》，《N5》に着目したアルゴリズムを提案する．《N1》のポートスキャンについては複数の端末への同一プロトコルでのアクセスを監視することで検知できる．《N4》のバックドアアクセスは通信相手から別のセッションが張られたことをもって検知する．また，《N5》のプログラム転送は，通常のプロトコルよりも大きいデータサイズの送信を検知することにする．以上の内容をアルゴリズムとして次にまとめた．

アルゴリズム (感染プロセスに着目したワーム検知アルゴリズム)

入力：・ポートスキャン台数 n_s [台]
・ポートスキャン監視時間 t_s [秒]
・バックドアアクセス監視時間 t_b [秒]
・プログラム転送監視時間 t_p [秒]
・プログラムとみなす
データの最小サイズ s [byte]

出力：・ワーム感染端末の IP アドレス

Step 1) ある端末 X について，t_s 秒以内の n_s 台以上の端末 $Y_1,Y_2,\ldots,Y_{ns}$ の特定ポート vp へのアクセスを検知する．

Step 2) Step1 で端末 X がアクセスした端末 Y_j ($1 \leqq j \leqq n_s$)のから, X のポート bp への新規セッション接続を検知する.
t_b 秒以上, 該当するアクセスがない場合は Step 1 へ戻る.

Step 3) 端末 X から端末 Y_j へ, サイズが s byte 以上となるデータが送信されたことを検知する (ただし,データのサイズとは, セッションの送信パケットに関する TCP, UDP データロードの合計サイズとする).
t_p 秒以上, 該当するアクセスがない場合は Step 2 へ戻る.

Step 4) 端末 X の IP アドレスを返す.

4. 検証

本提案アルゴリズムが, 従来の個々の挙動にのみ着目した検知方式と比較して, 通常トラフィックの誤検知（false positive）を減少させることを検証実験により確認した.

4.1. 検証実験内容

実際のオフィス環境で複数の Windows クライアント端末について, 通常業務日の一日に発生したパケットのキャプチャを行った(但し, TCP, UDP のパケットのみで, マルチキャスト, ブロードキャストを除く). キャプチャデータ毎に, 提案アルゴリズムに該当する通信の存在を調査した.

4.2. 実験条件

実験に用いたパラメータ, およびオフィス環境, 端末の種類を以下に示す.

パラメータ

- ポートスキャン台数 $n_s = 2$ [台]
- ポートスキャン監視時間 $t_s = \infty$[秒]
- バックドアアクセス監視時間 $t_b = \infty$[秒]
- プログラム転送監視時間 $t_p = \infty$[秒]
- 実行ファイルとみなす最小送信データサイズ $s = 500$[byte]

オフィス環境

- IP アドレス： 固定
- ドメイン： あり
- Web アクセス： プロキシサーバ経由
- メールサーバ： SMTP, POP3, IMAP4

端末

- 端末 1～4 : Windows 端末[有線 LAN 接続]
- 端末 5 : Windows 端末[無線 LAN 接続]
- 端末 6 : Linux 上の仮想 Windows

4.2. 結果と考察

実験結果を表 1 に示した. 表中の表記は次を意味する.

(C1) キャプチャ対象端末について, t_s 秒以内に n_s 台以上の IP アドレスの同一ポート vp へのアクセスが存在(ポートスキャン).

(C2) キャプチャ対象端末へ他の端末からのアクセスが存在(バックドアアクセスの有無).

(C3) キャプチャ対象端末から, 送信データサイズを s byte 以上とするような通信が存在(プログラム転送の有無).

(C4) キャプチャ対象端末と特定の端末との間で (C1)～(C3)が順番どおりに発生

上記の(C1)～(C4)は提案アルゴリズムの Step 1～Step 4 にそれぞれ対応している. それぞれに該当する通信が 1 回でも発生していた場合には“ ✔ ”を表示した. (C4)がチェックされている場合, 該当する端末が提案アルゴリズムによりワームに感染していると判断されたことになる.

表 1 : 検証実験の結果

	(C1)	(C2)	(C3)	(C4)
端末 1	✔	✔	✔	
端末 2	✔	✔	✔	✔
端末 3	✔	✔	✔	
端末 4	✔	✔	✔	✔
端末 5	✔	✔	✔	
端末 6	✔		✔	

端末 2 では誤検知を起こしているが, これはネットワークプリンタとのアクセスで, カラーとモノクロの 2 種類のプリンタにアクセスしたことで, スキャンとみなされたものであった. 今回の検証実験ではポートスキャンとみなすアクセス端末数 $n_s = 2$ としているために誤検知されたものである. n_s の値を大きくとることで, この誤検知は防ぐことができる. また, 端末 4 でも誤検知を起こしている

が，誤検知されたのは Windows ネットワークの名前解決に関するプロトコルで，WINS サーバとの通信によるものであった．それぞれのセッションには 4～5 分程度の差があった．通常，ワームはこれらの動作を非常に短時間で行うので，監視期間に関するパラメータ t_s, t_b, t_p の値を小さくとることで誤検知を防ぐことができると考えられる．また，双方ともにクライアントアクセスではなく，ネットワーク上の特別な役割をもった端末との通信である．通常のネットワークではネームサーバやネットワークプリンタの IP アドレスは固定であるので，これらの固定 IP 機器をフィルタリングすることでも対応可能と言える．

今回の検証実験で提案アルゴリズムによってワーム感染端末と見なされた通信は，ポートスキャンや監視時間，データサイズなどの閾値をかなり低く設定したにも関わらず，以上の 2 つのみであった．ポートスキャンなどの個々の現象単位での検知数の多さと比較して，順序どおりにこれらの現象が発生することはかなり少ないということが分かった．

以上のことから，ワームの個々の挙動だけでなくその順序も考慮することで誤検知を防げることが分かり，本提案方式の有効性を示すことができた．

6. むすびに

本稿では，ワームが感染するために必要とするプロセス(挙動とその順序)に着目したワーム検知方式を提案した．実際のオフィス環境における検証実験によって，ワームの個々の挙動だけでなくその順序も考慮することで誤検知を防げることが分かり，本提案方式の有効性を示した．

しかし，実際に感染拡大を防ぐには，感染プロセスの最後でワームプログラムが完全に転送される前に，感染端末を検知・隔離する必要がある．そのためには，ポートスキャンが検知された段階で，該当端末の通信をファイアウォール経由で行うように設定し，提案アルゴリズムの Step 3 でワームプログラムと判断された場合，即座に通信を遮断できる仕組みが必要であると考えている．今後はこのシステム構成を含めた，本提案方式の具体的な実現方法について検討していく．

参考文献

[1] “国内におけるコンピュータウィルス被害状況調査”，独立行政法人 情報処理推進機構 セキュリティセンター，http://www.ipa.go.jp/security/fy15/reports/virus-servey/documents/2003_virus_domestic.pdf, 2004 年 4 月．

[2] B. Mukherjee, L.T. Heberlein, and K.N. Levitt, "Network Intrusion Detection", *IEEE Network*, pp.26－41, May/June 1994.

[3] Snort, http://www.snort.org/

[4] P.A. Porras and P.G. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances", In Proceedings of the 20th National Information Systems Security Conference, pp. 353－365, October 1997.

[5] 馬場達也，鴨田浩明，小久保勝敏，松田栄之，“プロトコル仕様及びポリシー情報を利用した不正アクセス検知システムの実装と評価”，情報処理学会コンピュータセキュリティシンポジウム 2001（CSS2001）論文集，pp.173－178，2001 年 10 月．