# 送受信データ間の相関に基づく未知ワーム検知を利用した蔓延防止手法の提案

松本隆明[†] 杉村友幸[††] 鈴木功一[††]
前田秀介[†] 馬場達也[†]
水野忠則[††] 西垣正勝[††]

本論文ではコンピュータワームのネットワークを介した自己増殖機能に着目し，送受信データ間の相関を見ることにより未知ワームの検知を行う方法を提案する．本方式では，ネットワークを介して「外部から受信したデータ」と「外部に送信したデータ」が類似しているか否かを測ることによって，ネットワークを介したワームの自己増殖活動を捕える．よって，ある PC で未知ワームが新たに発見された場合，そのワーム自身を擬似的なウイルス定義ファイルとして利用することができる．これにより，本方式は未知ワームのエンタープライズネットワーク内への蔓延を効果的に抑止することが可能である．また，本方式は Winsock API をフックすることによりエンドユーザの PC における送受信データを常時監視するという実装が可能で，マスメーリング型およびネットワーク感染型の未知ワームのリアルタイム検知が可能である．検知率および処理コストを評価する基礎実験を行い，本方式の有効性を評価する．

# An Unknown-worm Spread Prevention Based on the Correlation between the Transmission and Reception Data

Takaaki Matsumoto,[†] Tomoyuki Sugimura,[††] Koichi Suzuki,[††]
Shusuke Maeda,[†] Tatsuya Baba,[†] Tadanori Mizuno[††]
and Masakatsu Nishigaki[††]

This paper proposes to detect unknown-worms by capturing self-propagation of worms over networks. In the proposed scheme, worm detection is carried out based on the correlation between the transmission and reception data. Therefore, the scheme is applicable to detect both unknown mass-mailing and network worms. In addition, once an unknown-worm is found by the scheme, the worm itself can be used as a kind of "virus definition file". This accomplishes to prevent the spread of any unknown-worm in an enterprise network. In this paper, the validity of the scheme is estimated with some basic experiments.

## 1. はじめに

近年のインターネットの普及にともない，（コンピュータワームを含む広義の）コンピュータウイルスによる被害は年々深刻なものとなってきている．経済産業省告示第 952 号「コンピュータウイルス対策基準」[1] 中のウイルスの定義によると，コンピュータウイルスとは，第三者のプログラムやデータベースに対して意図的に何らかの被害を及ぼすように作られたプログラムであり，(1) 自己伝染機能，(2) 潜伏機能，(3) 発病機能，の機能を 1 つ以上有するものとされている．なかでもワームと呼ばれる，自己複製を繰り返しながら破壊活動を行うプログラムによる被害は大きなものとなってきている[2),3)]．

また最近のワームは感染していくスピードがきわめて速く，2003 年に ATM ネットワークの障害や航空機予約システムの停止など全世界的に被害をもたらした Slammer ワームのケースでは，ワームの発生から 10 分以内に脆弱性を有するホストの 90%以上が感染したとの報告もある[22)]．これだけ感染のスピードが速いと，ワームの発生が報告されてからネットワーク内のホストに対処を行うのでは被害の拡大を抑えられないという問題が顕在化してきている．さらには，セキュリティホールが公開されると，当日のうちにその

† NTT データ技術開発本部
R&D Headquarters, NTT Data Corporation
†† 静岡大学情報学部情報科学科
Faculty of Informatics, Shizuoka University

脆弱性を悪用したワームが出現するという「ゼロデイアタック」といった攻撃も出現するようになり，アンチウイルスベンダの提供するウイルス定義ファイルの更新が間に合わないという問題も発生している．

現在のアンチウイルスソフトの主流となっているパターンマッチング法[5] に基づくワーム検出手法は，既知のワームを検出するには簡素で確実な手法ではあるが，最近のように感染スピードがきわめて速く，ゼロデイアタックを行うワームに対してはその効果が薄くなりつつあり，その対策には未知ワームを検知する技術が不可欠となってくる[4),6),7)]．

これまでに様々な未知ワーム検知手法が提案されてきているが，その代表的なものが，プログラムの振舞いにおける「ワームらしさ」を検知するビヘイビアブロッキング法[5] である．ワームらしい振舞いとしては，一般的に，レジストリの改ざん，システムファイルの書き換え，外部への感染活動などが規定される[5]．ビヘイビアブロッキング法は，プロセスが発行するシステムコールなどを検査することによりコンピュータ上で動作しているプログラムの動きを監視し，ワームらしい振舞いをした場合に，それをワームとして検知する．しかしビヘイビアブロッキング法には，ワームと類似した正常なプログラムを誤検知してしまう，また，多様化するワームの感染活動のすべてを実際に動作させてチェックすることは難しいという問題が残る．

静的ヒューリスティック法[8] は，プログラムを実行せずに，マシン語レベルのコード解析を行い，ワームらしい振舞いを検知する．ワームは実行されないので安全であるが，1 つの不正動作を行わせる際にもプログラムの書き方は多種多様であるので，コードを完全に解析することは容易ではない．すなわち，未知ワームを確実に検出することは難しい．

動的ヒューリスティック法[9] は，仮想環境でワームを実行させ，ワームらしい振舞いを検知する．ワームを実行して，実際にワームらしい振舞い（レジストリ改ざん，システムファイル変更，感染活動など）が発生したことにより検知を行うため，「基本的にはすべての未知ワームを検知することが可能であるが誤検知が多い」というビヘイビアブロッキング法とほぼ同じ特徴を持つ．また，潜伏期間が長いワームなどの場合，仮想環境下では動作せず，実環境で動作させたときに発病する可能性があるという問題を有する．さらに，エンドユーザのコンピュータ上で検出する場合には，VMware[10] などによる仮想環境を常駐的に稼動させるため CPU 負荷が大きすぎるといった問題も存在する．

このように，未知ワームを感染前に確実に検出することは負担が大きくかつ困難であるため，エンタープライズネットワーク内において，感染した端末からの 2 次感染や同一ワームによる蔓延を防ぐことが，企業活動の安全な遂行のためにはきわめて重要となる．

本論文では，「ネットワークを経由してコンピュータに侵入し，自己の複製をネットワークで送信する」というワームのネットワークを介した自己増殖機能に着目し，送受信データ間の相関を検査することによりワームによる感染を検知するとともに，新たに発見されたワーム本体を擬似的なウイルス定義ファイルとしてただちに使用することにより，エンタープライズネットワーク内で最初に感染したコンピュータを除き，他のコンピュータへのワームの蔓延を防止する手法を提案する．

## 2. 従 来 手 法

### 2.1 パーソナルファイアウォール

未知のワームからコンピュータを守るとともに，外部への波及を防止する手段としては，パーソナルファイアウォールが最も一般的な方法である．

パーソナルファイアウォールでは，ユーザの指示に従って，コンピュータから外部への通信ならびに外部からコンピュータへの通信を制御する．パーソナルファイアウォールを用いれば，ユーザが許可していないポートやプロトコルを用いた通信は遮断できる．また，アプリケーションごとに通信を遮断するかどうかを設定することも可能である．したがって，外部からのワームの侵入（1 次感染）を防ぐことができ，また，万一，コンピュータがワームに感染しても，他のコンピュータへさらに感染（2 次感染）しようとして外部へ通信を始める段階で阻止することができる．すなわち，未知のワームによる蔓延を食い止めることができる．

しかしながら，ユーザが許可しているポートやプロトコルあるいはアプリケーションを用いた通信は当然防ぐことはできないため，それらを利用して侵入してくるワームの 1 次感染を阻止することはできない．

また，2 次感染の際にも，感染したワームによる通信がパーソナルファイアウォールに検出された時点でユーザにアラートがあがるが，一般レベルのユーザにとってはそれがワームによる異常な発信なのか正常な発信なのかの識別が難しく，概して通信の許可を与えてしまいがちであるという問題も残る．

### 2.2 通信の挙動とプロセスの対応付けによる蔓延防止

未知ワームを検出して感染活動を停止させる方法として，通信の挙動とプロセスの対応をルール化し，あ

るプロセスからルールファイル上許されない通信が行われるとそれを検出して，当該プロセスを停止してワームの外部への感染を防止する方法が提案されている[23]．この方法は，シグネチャを用いないため，ワームの検出という点では未知ワームやゼロデイアタック攻撃にも有効であるとともに，感染が疑われるプロセスを停止することで，ワームのさらなる感染を防ぐという特長を持つ．

しかしながら，文献 23) の記述によれば，現時点では正常なプロセスになりすまして外部への通信を行うようなワームに対しては対処がなされていない．

また，本方式は，エンタープライズネットワーク内のあるコンピュータに感染した未知ワームが他のコンピュータに 2 次感染を試みる時点でこれを止めるものであり，未知ワームに感染したコンピュータがエンタープライズネットワークの外部にあり，そこからエンタープライズネットワーク内のコンピュータ群に次々と攻撃が仕掛けられるような場合には，その感染を防ぐことはできない．

## 3. 自己増殖の検出による未知ワーム検知・蔓延防止

本論文では，ワームが添付されたメールを大量に送信して感染を広げる「マスメーリング型ワーム」と，ネットワーク経由で脆弱性を狙って感染を広げる「ネットワーク感染型ワーム」の両方に対応可能な未知ワーム検知ならびに蔓延防止手法を提案する．

### 3.1 ネットワークを介したワームの自己増殖

ネットワークを介して自らの力で増殖を行うコンピュータワームの場合には，ネットワークを経由してコンピュータに侵入して，コンピュータ内で自己の複製を作成し，これを外部へ送りつけるという動作を繰り返す．

しかし，既存の未知ワーム検知手法においては，ワームの自己複製をワームらしい振舞いとして利用している例は，著者らが調べた限りでは存在していない．我々はその理由を以下のように考えている．

- ファイルコピーは利用者のコンピュータにおいて日常的に行われる操作であるため，ワームの自己複製と正常のファイルコピーを見分けることが難しい．
- ファイル単位でその複製を検知する方法は，ファイルに寄生するタイプの（狭義の）ウイルスに対しては無力である．

このため既存研究では，自己複製を検査する方法ではなく，プログラムを実行させた瞬間にネットワーク送信が発生するか否かをチェックしたり[13),14)]，利用者のコンピュータからの送信トラフィックが急に上昇するなどの異常を検出したりする[15)] というアプローチがとられている．しかしながら，前者の方法においては，正常にネットワーク通信を行うプログラムまで誤検出してしまうという問題を有し，後者の方法においても，一度に外部への感染を試みずに時間を空けて徐々に外部への感染を行うようなワームに対しては検出できないといった問題を有する．

これに対し，本論文では，ワームの「ネットワークを介した自己増殖機能」に着目し，送受信データ間の相関を検査することによりワームを検知する方法を提案する．具体的には，Windows OS の通信機能を司る Winsock API[12)] をフックすることにより利用者のコンピュータの送受信データを監視し，「外部から受信したデータ」と「外部に送信したデータ」が類似しているか否かを検査して，送受信データ間に類似したものがあった場合にワーム感染の疑いがあるとして検出する．

すなわち，単なる自己複製の検査ではなく，「外部からネットワークを介して利用者のコンピュータに侵入し，自分の複製をネットワークを通じて外部に発信する」という振舞いを検査することにより，正常なファイルコピーを誤検知するという問題を解消する．また，前述のような単なるネットワーク送信の発生有無による検知方法における誤検知の問題も避けることができる．

なお，この「ネットワークを介した自己増殖」という振舞いは，ネットワークを介して広まるコンピュータワームにのみ見られる特徴であり，（狭義の）ウイルスに対しては検知率を向上させるものではない．また，本方式は具体的には「受信データと送信データが十分に一致した」ことを検出する方法となるため，コンピュータに侵入するときと外部に感染するときでその姿が変異するポリモーフィック型ワームやメタモーフィック型ワーム☆を検知することは困難である．しかしながら，感染のたびに新しく暗号化鍵を生成し，自分自身を暗号化し直すタイプのワームであっても，ワーム自身を復号するためのプログラム部分は必ず平文であるはずであり，その部分を検出してマッチングすることにより，ワームを検出できる可能性がある．これらについては今後の検討課題である．

### 3.2 擬似定義ファイルによるワームの蔓延防止

本方式では，ネットワークを介して「外部から受信

☆ 文献 5) によれば，ポリモーフィック型ワームとは，感染のたびに異なる暗号化/復号ルーチンを生成して暗号化するワームを，また，メタモーフィック型ワームとは，感染のたびに異なる命令パターンを用いて自分自身を変更するワームのことを指す．

したデータ」と「外部に送信したデータ」が類似しているか否かを測ることによって，ネットワークを介したワームの自己増殖活動を捕える．よって，ある利用者のコンピュータで未知ワームが新たに発見された場合，以下のように，そのワーム自身を擬似的なウイルス定義ファイル（擬似シグネチャ）として利用することが可能である．

1. ある利用者のコンピュータで新たに検知された未知ワームを（必要に応じて無毒化処理を施したうえで）エンタープライズネットワーク内の他のコンピュータに渡す．
2. 未知ワームを受け取ったコンピュータは，これを擬似的に自身の「外部に送信したデータ」として登録する．
3. 未知ワームを受け取ったコンピュータは，その後，通常どおり「外部から受信したデータ」と「外部に送信したデータ」の類似度を定常的にチェックする．
4. 未知ワームを受け取ったコンピュータに実際に当該ワームが外部から侵入した時点で，3. のチェックの結果が陽性となり，当該未知ワームが検出される．

本方式ではこのように，エンタープライズネットワーク内の 1 台のコンピュータに未知ワームが感染した時点でこれを検出し 2 次感染を防ぐとともに，ただちに他のコンピュータに擬似定義ファイルを配布することが可能であるため，エンタープライズネットワーク内の 2 台目以降のコンピュータに感染する前に当該ワームをブロックすることができる．

これまで述べてきたように，本方式も含め現状の未知ワーム検知技術では，未知ワームを感染前に完全に検知することは不可能であるため，エンタープライズネットワーク内で最初のコンピュータ 1 台だけはワームに感染してしまうが，エンタープライズネットワーク内の他のコンピュータへのワームの蔓延が阻止できることは，企業や組織のネットワークにおいては業務の継続性が保たれるという意味で非常に重要なことである．本方式は，既存の技術に比べ，エンタープライズネットワーク内の感染コンピュータからの未知ワームの 2 次感染を阻止するだけでなく，外部にある感染コンピュータからエンタープライズネットワーク内のコンピュータ群に未知ワームが次々に送られてくることによる感染蔓延をも防止できるという特長を持つ．特に，アンチウイルスベンダの提供するウイルス定義ファイル（シグネチャ）が正式にリリースされる前に，とりあえず擬似シグネチャをエンタープライズネットワーク内に瞬時に配布することで疑わしい未知ワームをブロックできるというメリットの持つ意味は大きい．

### 3.3 エンドユーザのコンピュータにおけるリアルタイム検知

本方式では，ワームの自己増殖活動を捕えるために，単なる自己複製を検出するのではなく，「ネットワークを介して受信されたファイルが再び外部に送信される」という事象を監視することになる．これは，Windows OS の通信機能を司る Winsock API [12] をフックすることにより，エンドユーザのコンピュータにおける送受信データを常時モニタリングすることで達成される．よって，本方式は OS の DLL をラッピングするだけで実装が可能であり，マスメーリング型の未知ワームの検出はもちろん，エンドユーザのコンピュータでのリアルタイム検知が求められるネットワーク感染型の未知ワームの検出にも適応する．

ただし，「外部から受信したデータ」と「外部に送信したデータ」の類似度をチェックするというオーバヘッドは少なからず発生するため，その性能を評価する必要がある．本論文では 5 章でその評価を行う．

## 4. ワーム検知システムの実装

本方式は，Winsock API [12] をフックすることにより，エンドユーザのコンピュータにおける「外部から受信したデータ」と「外部に送信したデータ」をそれぞれのバッファにコピーし，両者の中に類似度の高いデータが存在するか否かをチェックする．

### 4.1 ラッパ DLL の実装

Winsock API の機能は，wsock32.dll および ws2_32.dll に内包されている [11]．今回の実装では，Windows 2000 を対象とし，ws2_32.dll のラッパ DLL を作成した．送信データの取得のために send, WSASend, sendto, WSASendTo, closesocket の 5 つの API を，受信データの取得のために recv, recvfrom, WSARecv, WSARecvFrom, closesocket の 5 つの API をフックする☆．

本ラッパ DLL により，送受信データの取得，および，データ間の類似度の検査を行う．ラッパ DLL の動作の詳細を以下に記す．開発環境には，Microsoft Visual C++ 6.0 を用いた．

### 4.2 データ受信時

ラッパ DLL を介して受信バッファにコピーされたデータは，ファイル単位で格納される．ファイル全体

☆ closesocket は送受信において同一の API なので，計 9 つの API をフックしている．

をすべてバッファリングする方法は，ストレージ容量の観点からもデータを比較する際のコストの観点からも非現実的であるため，効果的なデータ管理法が必要となる．本論文では，以下の 2 つの方法を試した．

(1) 分割ハッシュ比較法：

1. 各ファイルを先頭から 1,024 バイト単位で区切ったうえで，各データブロックをハッシュ化する．なお，ファイル $k$ のデータブロックの総数を $B_k$ とする．
2. 各ファイルに対するすべてのブロックのハッシュ値（以降，「ハッシュ値集合」と呼ぶ）を受信バッファに蓄える．ハッシュアルゴリズムは SHA-1 を採用しており，各ハッシュ値は 20 バイト長である．

(2) 部分データ比較法：

1. 各ファイルを先頭から 1,024 バイト単位で区切ったうえで，各データブロックの先頭 20 バイトを「部分データ」として取り出す．なお，ファイル $k$ のデータブロックの総数を $B_k$ とする．
2. 各ファイルに対するすべての部分データ（以降，「部分データ集合」と呼ぶ）を受信バッファに蓄える．

なお，受信バッファはすべてのプロセスで共有されるため，共有メモリ空間上に実装している．

### 4.3 データ送信時

ラッパ DLL は，Winsock API を介して送信されるデータを一時的に送信バッファに隔離する．そして，送信バッファ中のデータを，その時点までに受信バッファに格納されている各データ 1 つ 1 つと比較する．類似のデータがなければ，隔離していたデータを通信相手に送信する．送信データと類似度の高い受信データが検出された場合には，送信完了を意味する closesocket API を強制的に発行することにより通信先に通信のキャンセルを通知する．

なお，送受信データ間の類似度チェックは，受信バッファにおけるデータ管理方法に合わせて，以下のような 2 通りの方法での検査となる．

(1) 分割ハッシュ比較法：

1. すべての受信ファイル（受信バッファ内にハッシュ値集合が蓄えられているファイル）に対して，ハッシュ値が一致したデータブロック数をカウントするための変数を用意する．受信ファイル $k$ に対する変数を $z_k$ とする．すべての $z_k$ の値を 0 に初期化する．
2. 送信バッファ内に送信ファイルの先頭から 1,024 バイト分のデータ（送信ファイルの第 1 データブロック）がたまった時点で，この第 1 送信データブロックのハッシュ値を計算する．$i = 1$ として，以下の手順 3〜6 を実行する．
3. すべての受信ファイルのハッシュ値集合におけるそれぞれの第 $i$ ブロックのハッシュ値に対して，第 $i$ 送信データブロックのハッシュ値と一致するものがあるか否かを検査する．
4. 一致するものがなければ，第 $i$ 送信データブロックを実際に送信する．手順 7 に進む．
5. 第 $i$ 送信データブロックのハッシュ値が受信ファイル $k$ の第 $i$ ブロックのハッシュ値と一致した場合には，$z_k$ をインクリメントしたうえで，送信データと受信ファイル $k$ との一致度 $\frac{z_k}{B_k}$ を計算する．
6. $\frac{z_k}{B_k}$ がある閾値を超えたならば，送信完了を意味する closesocket API を強制的に発行することにより通信をキャンセルする．$\frac{z_k}{B_k}$ が閾値を超えていなければ，第 $i$ 送信データブロックを実際に送信する．
7. 送信バッファ内に次の 1,024 バイト分のデータ（送信ファイルの第 2 データブロック）がたまった時点で，この第 2 送信データブロックのハッシュ値を計算する．$i = 2$ として，手順 3〜6 を実行する．
8. 以下，送信バッファに次の 1,024 バイト分のデータがたまるごとに，$i$ をインクリメントしながら，手順 3〜6 を実行する．

(2) 部分データ比較法：

1. すべての受信ファイル（受信バッファ内に部分データ集合が蓄えられているファイル）に対して，部分データが含まれるデータブロック数をカウントするための変数を用意する．受信ファイル $k$ に対する変数を $z_k$ とする．すべての $z_k$ の値を 0 に初期化する．
2. 送信バッファ内に送信ファイルの先頭から 1,024 バイト分のデータ（第 1 送信データブロック）がたまった時点で，$i = 1$ として，以下の手順 3〜6 を実行する．
3. すべての受信ファイルの部分データ集合からそれぞれの $i$ 番目の部分データを取り出し，それと完全に一致するビット列が第 $i-1$ 送信データブロック☆および第 $i$ 送信データブロックの中に含まれるか（すなわち，各受信ファイルの 1,024$i$ バ

☆ $i = 1$ の場合に限っては，第 $i-1$ 送信データブロックは null データとなる．すなわち，各受信ファイルの先頭からの 20 バイト分のビット列と同じものが，送信ファイルの先頭から 1,024 バイト目までの中に含まれるかが検査される．

イト目から 20 バイト分のビット列が，送信ファイルの 1,024 $(i-1)$ バイト目から 1,024 $(i+1)$ バイト目までの中に含まれるか）を検査する．

4. 一致するものがなければ，第 $i$ 送信データブロックを実際に送信する．手順 7 に進む．
5. 第 $i-1$ 送信データブロックまたは第 $i$ 送信データブロックの中に受信ファイル $k$ の $i$ 番目の部分データが含まれていた場合には，$z_k$ をインクリメントしたうえで，送信データとファイル $k$ との一致度 $\frac{z_k}{B_k}$ を計算する．
6. $\frac{z_k}{B_k}$ がある閾値を超えたならば，送信完了を意味する closesocket API を強制的に発行することにより通信をキャンセルする．$\frac{z_k}{B_k}$ が閾値を超えていなければ，第 $i$ 送信データブロックを実際に送信する．
7. 送信バッファ内に次の 1,024 バイト分のデータ（第 2 送信データブロック）がたまった時点で，$i=2$ として，手順 3〜6 を実行する．
8. 以下，送信バッファに次の 1,024 バイト分のデータがたまるごとに，$i$ をインクリメントしながら，手順 3〜6 を実行する．

なお，分割ハッシュ比較法は各ブロックのハッシュ値をシグネチャとして，部分データ比較法は各ブロックの先頭 20 バイト分のデータをシグネチャとして，「ファイルの類似度をシグネチャベースのパターンマッチングで判定している」という意味では，上述のどちらの比較法も一般のアンチウイルスソフトで用いられているシグネチャによるワーム判定と類縁であるといえる．しかし，アンチウイルスソフトでは，専門家がワームプログラムの特徴的な一部分を抜き出すことによりシグネチャを事前に作成するものであるのに対し，本方式では，すべての受信データのシグネチャをリアルタイムに作成する必要がある．すなわち本方式では，シグネチャの生成に時間および人手をかけることはできない．このため，受信データと送信データを 1,024 バイトのブロックに分割し，各ブロックのシグネチャを機械的に作成し，ブロックごとにマッチング検査を実施している．

### 4.4 疑似定義ファイルによるブロッキング

3.2 節で述べたように，本方式においては，送信データと受信データを比較するというコンセプトによって，未知ワーム自身を擬似定義ファイルとして利用することが可能である．ただし，未知ワームの検知が「今までに受信したデータ」と「今から送信するデータ」の類似度を検査するのに対し，擬似定義ファイルによる当該ワームのブロッキングは「今，受信したデータ」と「今までに外部に送信したデータ（として格納されている擬似定義ファイル）」の類似度を測ることになる．

具体的には，送信バッファ，受信バッファに加え，定義ファイルバッファを用意したうえで，ラッパ DLL に以下の機能を実装する．

1. 他のコンピュータから届けられた擬似定義ファイルは，定義ファイルバッファに格納されている．
2. ラッパ DLL は，Winsock API を介して受信されるデータを，いったん受信バッファに隔離する．
3. ラッパ DLL は，受信バッファに隔離されたファイルを，定義ファイルバッファに格納されている各データ（擬似定義ファイル）1 つ 1 つと比較する．比較の方法には，4.2 節，4.3 節で説明した分割ハッシュ比較法，部分データ比較法などを用いればよい．
4. 類似のデータがなければ，受信データの隔離を解除する．擬似定義ファイルと類似度の高いデータの受信が確認されたならば，隔離した受信データを廃棄する．

## 5. 実　　験

本方式の実現の可能性を示すために，4 章のラッパ DLL を実装し，実際にワームの検知を行った．ただし，未知ワームの入手が不可能であるため，ここでは 4 章の方式によって代表的な既知ワームを検出することが可能であるかを測定することとした．また本論文においては，試行的に，両比較法ともデータの一致度 $\frac{z_k}{B_k}$ の閾値を 0.9 とした．すなわち，データを送信する際に，そのデータと 90%以上類似しているデータがすでに受信されている場合に，当該データを「ワームの疑いあり」と判断し，送信をキャンセルする．

### 5.1 実験環境

本実験で使用したネットワーク構成を**図 1** に示す．サーバ PC のスペックは OS：Fedora Core 2，CPU：Pentium4 1.6 GHz，メモリ：384 MB，マシン A の PC のスペックは OS：Windows 2000，CPU：Pentium2 400 MHz，メモリ：128 MB，マシン B の PC のスペックは OS：Windows 2000，CPU：Pentium2 400 MHz，メモリ：384 MB である．

サーバには DNS 名が割り当てられており，ドメイン名が infected.net，ルータ，DNS サーバ，Mail サーバはそれぞれ gw，ns，mail という名前である．ただし，実際にはルータ，DNS サーバ，Mail サーバは 1 台の PC で実装している．サーバ PC の OS は Fedora

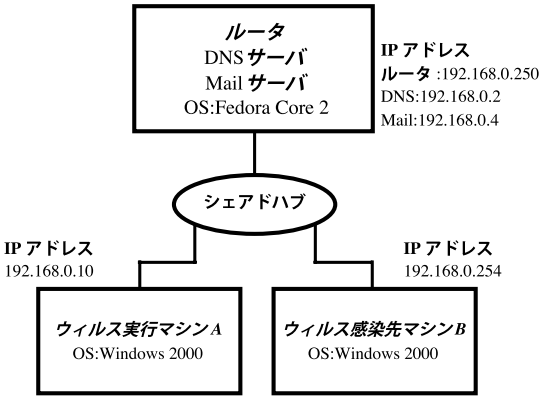


**図 1** 隔離ネットワーク構成図
Fig. 1 Experimental network.

**表 1** ワーム検知実験の結果
Table 1 Worm detection by the proposed scheme.

| 名前 | 型 | 分割ハッシュ比較 | 部分データ比較 |
|---|---|---|---|
| Beagle.X | マスメーリング型 | ○ | ○ |
| Mydoom.Q | | ○ | ○ |
| Sobig.F | | ○ | ○ |
| Netsky.B | | ○ | ○ |
| Netsky.D | | ○ | ○ |
| Blaster.C | ネットワーク感染型 | × | ○ |
| Sasser.C | | × | ○ |

Core 2 である．

クライアントの PC は 2 台であり，すでにワームに感染しているマシン A がワームには未感染のマシン B に攻撃をしかける．マシン A もマシン B も OS はセキュリティパッチが当たっていない Windows 2000，メーラは Outlook Express である．なお，@infected.net ドメイン宛てのメールはすべてマシン B のメーラにて受信可能なように設定した．そして，マシン B に 4 章で示したラッパ DLL を装備した．

### 5.2 ワーム検知実験

#### 5.2.1 マスメーリング型ワームの検知

マシン A のワームを実行し，マシン B にワームメールを送信させる．マシン B は，マシン A からのワームメールを受信し，メールの添付ファイル（ワーム本体）をユーザが故意に実行することにより自らもワームに感染し，ワームメールを発信し始める．代表的な既知ワームの感染に対し，マシン B に装備されているラッパ DLL が，これを検知できるかどうか確かめた．

#### 5.2.2 ネットワーク感染型ワームの検知

マシン A のワームを実行し，マシン B に対してマシン B の脆弱性を突いた攻撃を行わせる．マシン B は，マシン A からの攻撃を受け，自らもワームに感染し，他のマシンを攻撃し始める．代表的な既知ワームの感染に対し，マシン B に装備されているラッパ DLL が，これを検知できるかどうか確かめた．

#### 5.2.3 実験結果と考察

マスメーリング型ワーム，ネットワーク感染型ワームともに，分割ハッシュ比較法，分割データ比較法により検知を行った．それぞれの手法による検知結果を**表 1** に示す．本方式により検知できなかったワームに対してのみ，以下にその理由を示す．

Blaster.C，Sasser.C は，先頭数バイトに通信用制御コードが含まれており，それ以降がワーム本体となっていた．よって，分割ハッシュ比較法では検出できなかった．

難読化や暗号化によって自分自身を変異させていくポリモーフィック型ワームやメタモーフィック型ワームへの対応は今後の課題であるが，3.1 節で述べたように，感染のたびに新しく暗号化鍵を生成し，自分自身を暗号化し直すタイプのワームについては，ワーム自身を復号するためのプログラム部分は必ず平文であるはずであり，その部分を検出してマッチングすることにより，ワームを正しく検出できる可能性がある．また，本方式をアプリケーション層レベルで実装することが可能であれば，いったん復号してからマッチングを行えば検出できる可能性がある☆．

今回の検知実験の結果によれば，分割ハッシュ比較法による比較を行うだけで不変異型のワームのほとんどを検知することができ，部分データ比較法による比較によれば不変異型のワームをすべて検知することができた．これにより，ワームのネットワークを介した自己増殖をワームらしい振舞いとして検出する本方式の有効性が確認できた．

### 5.3 従来方式との比較

本方式の目的は，エンタープライズネットワーク内において，感染した端末からの 2 次感染や同一ワームによる外部からの連続攻撃から，エンタープライズネットワークを守ることである．したがって，安全側に倒してエンタープライズネットワークを守ることを基本的な考え方としている．しかしながら，疑わしきはワームという方向に倒しすぎると，適用される環

☆ ただし，自身を暗号化しているワームをいったん復号するということは，通常，プログラム本体を実行することにつながるため，ワームを起動せずに復号だけを行う工夫が必要となる．

境によっては従来方式に比べて誤検知のケースが増加して実運用上問題となる場合があるため，その評価を行う．ここでは，本方式と既存のビヘイビアブロッキング法について，検知漏れと誤検知の発生状況を実測する．

評価に際し，5.2 節の実験で用いたワームのほかに，いくつかの一般的なアプリケーションを用意した．用意したアプリケーションは，常駐型アプリケーションのインストーラ（memcl[16]），メーラ（Edmax），FTP クライアント（Smart FTP），ブラウザ（Internet Explorer），Ethereal，アンチウイルスソフト☆（Norton AntiVirus）のインストーラである．これらに対し既存のビヘイビアブロッキング法で規定されている「ワームらしい振舞い」と本方式による検知を実際に行い，評価を行った．

本実験では，既存のビヘイビアブロッキング法で規定されている「ワームらしい振舞い」としては以下の 5 つをとりあげた．それぞれの説明と具体的な監視方法，および，誤検知，検知漏れの評価結果を以下に示す．また，比較のために本方式（ここでは，5.2 節の実験において検知率が高かった部分データ比較法を採用した）についても示す．

1. タイプ 1：OS 起動時の自動実行[17]
   【振舞い】：ワームはできるだけ長い期間，クライアントに感染し続けようとするため再起動後にも自身が自動実行されるようにする．
   【監視対象】：OS の自動実行に関わるレジストリや，スタートアップへの書き込み（タイプ 1）．
   【評価】：常駐型のプログラムは OS 起動時に自動実行するためレジストリの変更を行う必要がある．よって，インストーラをワームであると誤検知してしまう．
2. 起動直後のファイルコピー[18]
   【振舞い】：ワームは OS のシステムフォルダ以下に自身のコピーの書き込みを試みる．
   【監視対象】：システムフォルダ以下への書き込み（タイプ 2）．
   【評価】：インストーラは起動直後に OS のシステムフォルダ以下にファイルコピーを行った．よって，インストーラをワームであると誤検知してしまう．
3. 別プロセスの起動[19]
   【振舞い】：ワームは別のプロセスを起動することによって自身の行動を隠そうとする．また，感染などの目的のために別のプロセスを起動する．
   【監視対象】：ユーザが起動していないプロセスの追加（タイプ 3）．
   【評価】：ブラウザとアンチウイルスソフトのインストーラは別プロセスを起動した．よって，これらをワームであると誤検知してしまう．一方，ワームの中に別プロセスを起動しないものが存在した．このようなワームに対しては検知漏れが生じる．
4. トラフィック量の上昇[20]
   【振舞い】：ワームは他の多数の PC への自己複製を行う．
   【監視対象】：PC の送信トラフィック量の上昇（タイプ 4）．
   【評価】：FTP クライアントでは，FTP 通信時にトラフィックが上昇した．ネットワークを利用するアプリケーションは，実行時に少なからずトラフィックが上昇する☆☆．このようなアプリケーションに対して，トラフィックがどれくらい増加したらワームであるかの閾値を適切に求めることは一般的に難しいため，閾値を低く設定してしまうと誤検知が，閾値を高く設定してしまうと検知漏れが発生する可能性がある．
5. CPU 利用率の上昇[21]
   【振舞い】：ワームは侵入，発病，感染の動作をなるべく速く行おうとする．
   【監視対象】：プロセスの CPU 利用率の上昇（タイプ 5）．
   【評価】：Ethereal の実行時には CPU 利用率が上昇した．多大な計算処理をともなうアプリケーションにおいては，CPU が寡占されることも多い．このようなアプリケーションに対して，CPU 利用率がどれくらい増加したらワームであるかの閾値を適切に求めることは一般的に難しいため，閾値を低く設定してしまうと誤検知が，閾値を高く設定してしまうと検知漏れが発生する可能性がある．
6. 本方式
   【振舞い】：ワームは PC への感染後，他の PC へネットワークを介して自己複製を行う．
   【監視対象】：「外部から受信したデータ」と「外部へ送信するデータ」の相関．

☆ アンチウイルスソフトも常駐型アプリケーションのカテゴリに含まれるが，ここでは特に個別に評価した．

☆☆ メーラにおいては，送受信を行ったメールの本数やファイルサイズが多大ではなかったため，誤検知には至らなかった．

**表 2** 誤検知および検知漏れの評価表
Table 2 Estimation results.

| | ワームらしい振舞い（監視対象） | | | | | |
|---|---|---|---|---|---|---|
| | タイプ 1 | タイプ 2 | タイプ 3 | タイプ 4 | タイプ 5 | 本方式 |
| インストーラ（常駐型） | 誤検知 | 誤検知 | × | × | × | × |
| メーラ | × | × | × | × | × | 誤検知 |
| FTP | × | × | × | 誤検知 | × | × |
| ブラウザ | × | × | 誤検知 | × | × | × |
| Ethereal | × | × | × | × | 誤検知 | × |
| インストーラ（アンチウイルス） | 誤検知 | 誤検知 | 誤検知 | 誤検知 | × | × |
| ワーム | ○ | ○ | 検知漏 | 検知漏 | 検知漏 | ○ |

【凡例】×：非検知，○：検知

【評価】：ユーザが自分宛に届いたメールを友人などに転送する際に誤検知が発生する．

以上の結果をまとめたものが**表 2** である．表 2 より，それぞれの「ワームらしい振舞い」を個別に見た場合には，本方式は，他の振舞いを監視する方法と比べて，検知漏れが起こりにくく，かつ，誤検知が少ない方法であるということが分かる．

しかしながら，オフィス業務のようにメールの転送が日常茶飯事的に発生するエンタープライズネットワーク環境においては，本方式の誤検知発生率は相当のものになることが予想される．このためこうした環境においては，本方式を既存方式と併用することにより誤検知の発生を防止する．表 2 の結果から，たとえばタイプ 1 の方式と本方式を以下のように併用する．

- タイプ 1 の方式と本方式の両方でアラートが上がり，かつ本方式で Winsock API を呼び出したプロセス名とタイプ 1 の方式で検知したプロセス名とが同一である場合，ワームであると判別する．

これにより，たとえばユーザが正規のメール転送と，あるソフトウェアのインストールを同時に行った場合，タイプ 1 の方式と本方式の両方でアラートが上がることになるが，メールの転送を行ったプロセス（メーラ）とレジストリ書き換えを行ったプロセス（インストーラ）とは別のプロセスとなるためワームとして誤検知されることはない．ただし，正規のメーラの送信機能を利用してワームが添付されたメールを送信するようなワームの場合には，メールの転送を行ったプロセス（メーラ）とレジストリ書き換えを行ったプロセス（ワーム）が異なるため，上記手段では検知できない．そこで，さらにタイプ 4 の方式を組み合わせ，プロセスが異なる場合でも，一定時間内に送信されたメールの数がある閾値を超えた場合は，ワームであると判断することで検知することが可能となる．

以上のように，メールの転送が定常的に行われるようなエンタープライズネットワーク環境では本方式と既存方式を併用することで，メール転送を誤検知することなくワームを検知することができる．

もちろん，Slammer ワームの際に問題となった ATM ネットワークやオンライン予約サービスをはじめとする重要インフラなどに見られるような，エンタープライズネットワーク内のクライアント間でメール転送が生じないような環境下では，本方式だけの適用であっても十分に有用な方式である．

### 5.4 オーバヘッド

ラッパ DLL が行う送受信ファイル間の相関チェックには相応のオーバヘッドが見込まれる．そこで，ws2_32.dll のラッパ DLL を用いた際のエンドユーザの PC における処理時間のオーバヘッドを測定した．エンドユーザの PC のスペックは，OS：Windows 2000, CPU：Pentium4 2.8 GHz, メモリ：512 MB である．

今回は，ブラウザ（Mozilla Firefox 1.0）によるネットサーフィンおよびメーラ（Becky! Internet Mail 2）によるメールの送受信を行う際のオーバヘッドを測定した．まず，あるユーザにエンドユーザの PC にて，およそ 2 時間にわたって約 300 サイトへのネットサーフィンと，約 10 通のメールの受信を行ってもらった．その間に受信バッファに格納されたファイル数は 445 個（18,911,428 バイト）であり，1 秒あたり約 2,626 バイトの受信量であった．そしてそのうえで，任意のサイトに Web アクセスをし，134,735 バイトのファイルを受信した際のオーバヘッドと，2,129,838 バイト（Base64 エンコード後は 2,914,516 バイト）のファイルを添付したメールを送受信した際のオーバヘッドを測定した．その結果を**表 3** に示す．表 3 中の各項の意味は以下のとおりである．

データ受信時：ファイルを受信してから，ファイルのハッシュ値集合／部分データ集合を受信バッファに格納するまでの時間（ms）
データ送信時：ファイルを送信する際に，受信バッファに格納されているすべてのハッシュ値集合／部分データ集合との比較を行うのに要した時間（ms）
ファイル数：受信したファイル数
データ数：受信バッファに格納されたハッシュ値また

**表 3** オーバヘッドの測定結果
Table 3 Overheads.

| | データ受信時 (ms) | | データ送信時 (ms) | | ファイル数 | データ数 |
|---|---|---|---|---|---|---|
| | ブラウザ | メーラ | ブラウザ | メーラ | | |
| 分割ハッシュ比較 | 47 | 264 | 93 | 940 | 445 | 18697 |
| 部分データ比較 | 47 | 217 | 125 | 1880 | | |

は部分データの総数

比較手法によってオーバヘッドは異なるが，ネットサーフィンにおいては最大で 130 ms 程度のオーバヘッドであるので，ユーザの感じるストレスはほとんどないと思われる．メーラにおけるデータ送信においては，およそ 2 M バイト（Base64 エンコード後はおよそ 3 M バイト）ものメールに対して類似度の検査が行われるにあたって，さすがに 2 秒程度のオーバヘッドが生じている．しかし，実効速度 1 Mbps のネットワーク上で約 3 M バイトのメールを送受信するのにおよそ 23 秒を要することを鑑みると，本方式を実装した場合のファイルの送信時間のオーバヘッドは約 8%（$\frac{2\text{秒}}{23\text{秒}+2\text{秒}}$）にすぎない．また，今回の実装はプロトタイプ段階にとどまっており，今後，送受信データ比較のアルゴリズムの検討を行うことでさらなる高速化が達成される余地は残されている．

以上より，本方式は基本的にはオーバヘッドが大きいものの，過去 2 時間程度の受信ファイルを保持して，データを送信するたびに送信ファイルと受信ファイルの比較を行ったとしても，パフォーマンスの劣化はさほど高くはならないことが確認できる．

## 6. 疑似定義ファイルの効果

3.2 節で述べたように，本方式では，LAN の中の 1 台目の PC における未知ワームの感染が検出された時点で，未知ワーム自身を疑似定義ファイルとして他の PC に配布することが可能である．すなわち，LAN 内の 2 台目以降の PC においては，当該ワームの侵入時に，これをブロックすることができる．本章では，本方式が提供する疑似定義ファイルの有効性を検証する．

### 6.1 送受信データ比較方式と擬似定義ファイル方式の併用

理想的には，LAN 内のすべての PC が本方式を実装し，送受信データの比較を常時行うことが望ましい．LAN 内のすべての PC が本方式を実装していれば，未知ワームが LAN の中の 1 台目に感染した時点でこれが検知され，他の PC に擬似定義ファイルが配布される．すなわち，LAN 内の PC の総数を $N$，1 台の PC がワーム被害にあった場合の損害額の平均値を $D$ とすると，未知ワームに感染してしまう PC は $N$ 台中の最初の 1 台のみであり，未知ワームによる被害額の期待値は LAN 全体で $D$ に抑えられる．

しかし，5.4 節でも述べたように，本方式ではつねに送信データと受信データの類似を検査し続ける必要があり，相応のオーバヘッドが発生する．このため，LAN 内に 1 世代前の PC が存在しているような場合などに，CPU パワーが低い PC に本方式を実装することが躊躇されることもありうるかもしれない．このような場合に有効であると考えられるのが，本方式の擬似定義ファイルのみを活用する方法である．すなわち，4.4 節に示した機能のみをラッパ DLL に実装する．以降では，両者を区別するために，すべての送受信データの類似度検査を実装した方式を「送受信データ比較型」，擬似定義ファイルの検査のみを実装した方式を「擬似定義ファイル型」と呼び分けることにする．

ここでの被害額とは，簡単のため，その PC が提供するサービスがストップすることによるビジネス機会損失や復旧に要する費用などを包含したものを指すこととする．実際には，1 台でもエンタープライズネットワーク内での PC が感染すると顧客への賠償や企業イメージのダウンなどによる損失も発生するが，ここでは損害額を厳密に見積もることが目的ではなく，擬似定義ファイルを用いた未知ワームの検出方式がコスト的に意味があるかどうかを検証することが目的であるので，モデルを簡単化する．

受信データを擬似定義ファイルと照合するという作業が発生するのは，ある PC に未知ワームが侵入して他の PC に擬似定義ファイルが配布されてからアンチウイルスベンダによって正規のウイルス定義ファイルがリリースされるまでの間に限られ，通常時には行われない．よって，送受信データ比較方式のオーバヘッドを $C_1$，擬似定義ファイル方式のオーバヘッドを $C_2$ とした場合，

$$C_1 \gg C_2 \quad (1)$$

$$C_2 \approx 0 \quad (2)$$

が成立する．

そこで本章では，以降，LAN 内に送受信データ比較方式を実装している PC と擬似定義ファイル方式を実装している PC が混在している場合を想定して，両方式の導入率とその効果に関して検討していく．ここ

で，LAN 内で送受信データ比較方式を実装している PC の割合（%）を $x$（すなわち，擬似定義ファイル方式を実装している PC の割合は $1-x$）とする．

### 6.2 両方式の導入比率

LAN 内で送受信データ比較方式を実装している PC の数 $n_1$ は

$$n_1 = xN \tag{3}$$

であり，擬似定義ファイル方式を実装している PC の数 $n_2$ は

$$n_2 = (1-x)N \tag{4}$$

である．

擬似定義ファイル方式を実装している $n_2$ 台の PC は未知ワームを新たに発見する機能はないので，新種の未知ワームが LAN 内に侵入した場合，当該ワームが送受信データ比較方式を実装している $n_1$ 台の PC のうちのいずれかに感染した時点で，それが検出されることになる．すなわち，確率的には，当該ワームが LAN 内で $m$ 台の PC に感染した時点で，それが検知されると期待される．ここで，

$$m = \frac{N}{n_1} = \frac{1}{x} \tag{5}$$

である．

未知ワームが送受信データ比較方式を実装している PC に発見された直後に，擬似定義ファイルが生成され，LAN 内のすべての PC に配布される．よって，これ以降，LAN 内の PC が当該ワームに感染することはない．ここで，擬似定義ファイルの生成および配布は機械的に実行可能なため，それに要する時間は無視できるとすると，結局，この時点までにおいて当該ワームに感染した LAN 内の PC は $m$ 台であり，その損害額 $f_1(x)$ は次式となる．

$$f_1(x) = mD = \frac{D}{x} \tag{6}$$

一方，LAN 内の $n_1$ 台の PC においては送信データと受信データを常時比較するために $C_1$ のオーバヘッドが，$n_2$ 台の PC においては受信データと擬似パターンを比較するため $C_2$（$\approx 0$）のオーバヘッドが発生している．このオーバヘッドは PC の作業効率の悪化として現れるため，これも一種の損害であると算定すると，その損害額 $f_2(x)$ は LAN 全体で次式のように導かれる．

$$\begin{aligned} f_2(x) &= n_1C_1 + n_2C_2 \\ &= NC_1x + NC_2(1-x) \end{aligned} \tag{7}$$

以上より，LAN 内で送受信データ比較方式を実装する PC の割合 $x$ については，全体の損害

$$f(x) = af_1(x) + bf_2(x) \tag{8}$$

を最小にするように設定するのが最も効率的であるということが分かる．なお，式 (8) の $a$ と $b$ は適当な加重係数である．

簡単のために，$a:b=1:1$ として計算してみると，擬似パターン比較方式のオーバヘッド $C_2$ はほぼゼロであるとしたので，式 (8) は

$$f(x) = \frac{D}{x} + NC_1x \tag{9}$$

となる．式 (9) を解析すると，

$$f'(x) = -\frac{D}{x^2} + NC_1 = 0 \tag{10}$$

を満たす点で $f(x)$ が最小値を得ることが分かる．

$0 \le x \le 1$ であるので，その解は，

$$x = \sqrt{\frac{D}{NC_1}} \tag{11}$$

である．

### 6.3 考　　察

1 台の PC がワームに感染した際の損害額や，送受信データ比較方式や擬似定義ファイル方式のオーバヘッドが関与する損害額を見積もることは，実際には非常に難しい問題であるのだが，式 (11) の結果は，

- Case 1：$C_1 \gg D$ のとき

  $$x = \sqrt{\frac{D}{NC_1}} \approx 0$$

  よって，「ワームに感染した際の損害額が小さければ，送受信データ比較方式を導入する必要はなく，すべての PC に擬似定義ファイル方式を実装すれば十分である」ということを意味する．ただし，この場合は擬似定義ファイルを生成する PC が存在しないため，LAN の外部から擬似定義ファイルの供給を受ける必要性がある．
- Case 2：$C_1 \ll D$ のとき

  $$x = \sqrt{\frac{D}{NC_1}} \approx \infty$$

  $0 \le x \le 1$ より，$x = 1$ である．よって，「ワームに感染した際の損害額が大きい場合には，相応のオーバヘッドがあろうとも，すべての PC に送受信データ比較方式の未知ワーム検知を導入する必要がある」ということを意味する．
- Case 3：$C_1 = D$ のとき

  $$x = \sqrt{\frac{D}{NC_1}} = \sqrt{\frac{1}{N}}$$

  上記の式より，$n_1 = xN = \sqrt{N}$ となる．また，$m = 1/x = \sqrt{N}$ である．よって，データ比較方式のオーバヘッド（本方式の導入コスト）とワームに感染した際の損害額が等価である場合には，$N$ 台

の PC の中の $\sqrt{N}$ 台に送受信データ比較方式を導入すると効率が良い」ということを意味する．

$C_1 = D$ のときを例にとると，たとえば LAN 内の PC が 50 台である場合，送受信データ比較方式を導入すべき PC は 7 台であり，残り 43 台は擬似定義ファイル方式でよい．そして，損害が生じる PC（擬似定義ファイルが生成されるまでに未知ワームに感染する可能性がある PC）は平均で 7 台となる．また，LAN 内の PC が 1,000 台である場合であっても，送受信データ比較方式を導入すべき PC は 32 台であり，残り 968 台は擬似定義ファイル方式でよい．そして，損害が生じる PC は平均で 32 台となる．よって，（ワーム被害の際の損害額や本方式のオーバヘッドが関与する損失を非常におおまかに見積もった限りでは）大規模な LAN 上であっても，少ない台数の PC にオーバヘッドの大きい送受信データ比較方式を導入することで，効果的に未知ワーム対策を展開することが可能であるという示唆が得られた．

## 7. ま と め

ワームのネットワークを介した自己増殖機能に着目し，送受信データ間の相関を見ることにより未知ワームを検知し蔓延防止を行う方法を提案した．検知率および処理コストを評価する基礎実験から，本方式の有効性が確認できた．

本方式では，ネットワークを介して「外部から受信したデータ」と「外部に送信したデータ」が類似しているか否かを測ることによって，未知ワームを検知する．よって，あるコンピュータで未知ワームが新たに発見された場合，そのワーム自身を擬似的なウイルス定義ファイルとして利用することができるため，エンタープライズネットワーク内で最初に感染したコンピュータを除き，他のコンピュータへのワームの蔓延を防止することができる．

また，本方式は，Winsock API をフックすることによりエンドユーザの PC における送受信データを常時監視するという実装が可能であり，マスメーリング型およびネットワーク感染型の未知ワームのリアルタイム検出が可能である．

本方式をポリモーフィック型やメタモーフィック型の未知ワームへも対応させることが今後の課題である．

## 参 考 文 献


1) 経済産業省：告示第 952 号，コンピュータウイルス対策基準．http://www.meti.go.jp/policy/netsecurity/CvirusCMG.htm
2) アットマーク・アイティ：Code Red ワームの正体とその対策［改訂版］．http://www.atmarkit.co.jp/fwin2k/insiderseye/20010803codered/codered.html
3) アットマーク・アイティ：ネットを震撼させたコンピュータ・ワーム，Nimda を検証する．http://www.atmarkit.co.jp/fwin2k/insiderseye/20010922nimda/nimda_01.html
4) INTERNET Watch：“ゼロデイアタック” の脅威が増している．http://internet.watch.impress.co.jp/cda/special/2004/03/19/2498.html
5) 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
6) ITmedia：マイクロソフトの奥天氏，ウイルスをめぐる状況は「悪化している」．http://www.itmedia.co.jp/enterprise/articles/0407/23/news020.html
7) INTERNET Watch：シマンテックに聞く「ウイルス被害の傾向と対策」．http://internet.watch.impress.co.jp/www/article/2003/0409/symantec.htm
8) Computer Associates/Malivanchuk, T.: The Win32 worms: classification and possibility of heuristic detection. http://www.virusbtn.com/conference/vb2002/abstracts/heuristic.xml
9) ESET: NOD32/ヒューリスティックスキャンエンジン．http://www.mediaselect.co.jp/magazine/its/0302/0302210071.html
10) Vmware. http://www.vmware.com/
11) 澤川　渡，綱島明宏：TCP/IP 解析とソケットプログラミング，オーム社．
12) IT 用語辞典 e-Words, “Winsock とは”．http://e-words.jp/w/Winsock.html
13) 神薗雅紀，白石善明，森井昌克：仮想ネットワークを使った未知ウィルス検知システム，電子情報通信学会技術報告，ISEC2003-27, pp.113–120 (2003).
14) 中谷直司，小池竜一，厚井裕司，吉田等明：メール型ウィルス感染防御ネットワークシステムの提案，情報処理学会論文誌，Vol.45, No.8, pp.1908–1919 (2004).
15) Snort: The Open Source Network Intrusion Detection System. http://www.snort.org/
16) メモリの掃除屋さん．http://www6.plala.or.jp/amasoft/soft/soft1/memcl.html
17) シマンテック Carey Nachenberg: Behavior Blocking: The Next Step in Anti-Virus Protection/ビヘイビアブロッキング．http://www.securityfocus.com/infocus/1557
18) クワンタム・リープ・イノヴェーションズ・インコーポレーテッド/シュヌラー：コンピュータ･ウィルス・トラップ装置，特表平 10-501354 (1998).
19) Japan Network Security Association Dynamic

Defense Working Group：ホストベースの IDS の概要と適用について．http://www.jnsa.org/active/houkoku/IDSBasic.pdf
20) 株式会社東芝/高橋俊成：コンピュータウィルス発生検出装置，方法，およびプログラム，特開 2003-241989 (2003).
21) 東日本電信電話株式会社/鈴木 晃：電子メール中継システム及び電子メール中継方法，特開 2002-314614 (2002).
22) Moore, D., et al.: The Spread of the Sapphire/Slammer Worm. http://www.cs.berkeley.edu/~nweaver/sapphire
23) 面 和成，東角芳樹，鳥居 悟，武仲正彦：セキュアな企業内ネットワークの実現—メールウィルスによるゼロデイアタックの防御，コンピュータセキュリティシンポジウム 2004 論文集，Vol.I, pp.19–24 (2004).

(平成 17 年 9 月 1 日受付)
(平成 18 年 3 月 2 日採録)

**松本 隆明**（正会員）

1978 年東京工業大学大学院電子物理工学専攻修士課程修了．同年日本電信電話公社（現 NTT）入社．現在，株式会社 NTT データ技術開発本部本部長．オペレーティングシステム，コンピュータアーキテクチャ，情報セキュリティに関する研究に従事．静岡大学客員教授．

**杉村 友幸**

2003 年静岡大学情報学部情報科学科卒業．2005 年静岡大学大学院修士課程修了．同年株式会社富士通ソーシアルサイエンスラボラトリ入社．在学中，情報セキュリティに関する研究に従事.

**鈴木 功一**

2005 年静岡大学情報学部情報科学科卒業．現在，静岡大学大学院修士課程．情報セキュリティに関する研究に従事.

**前田 秀介**（正会員）

2004 年電気通信大学大学院電気通信学研究科情報工学専攻博士前期課程修了．同年株式会社 NTT データ入社．現在，同社技術開発本部にてネットワークセキュリティに関する研究に従事.

**馬場 達也**（正会員）

1995 年慶應義塾大学理工学部電気工学科卒業．同年 NTT データ通信株式会社（現，株式会社 NTT データ）入社．現在，同社技術開発本部にてネットワークセキュリティに関する研究に従事．著書に『マスタリング IPsec』（オライリー・ジャパン）がある．IEEE 会員.

**水野 忠則**（フェロー）

1945 年生．1968 年名古屋工業大学経営工学科卒業．同年三菱電機株式会社入社．1993 年静岡大学工学部情報知識工学科教授，現在，情報学部情報科学科教授．工学博士．情報ネットワーク，モバイルコンピューティング，放送コンピューティングに関する研究に従事．著訳書としては『コンピュータネットワーク概論』（日経 BP），『モダンオペレーティングシステム』（ピアソン・エデュケーション）等がある．電子情報通信学会，IEEE，ACM 各会員．当会フェロー，監事.

**西垣 正勝**（正会員）

1990 年静岡大学工学部光電機械工学科卒業．1992 年静岡大学大学院修士課程修了．1995 年同大学院博士課程修了．日本学術振興会特別研究員（PD）を経て，1996 年静岡大学情報学部助手．1999 年同講師，2001 年同助教授．博士（工学）．情報セキュリティ，ニューラルネットワーク，回路シミュレーション等に関する研究に従事.