# ウイルス・ワーム感染パターン解析

清野祥之* 森口一郎**

仮想環境内でウイルス・ワームの感染速度やアクセスパターンを収集し、解析・視覚化を行うシステムを構築した。この解析システムでは、ウイルスの攻撃先IPアドレスの変化に相関が存在するかを解析するためのプロット図を表示でき、さらに、攻撃先の推移を動画で視覚的に表現することもできる。また、本解析システムは仮想環境を利用するため、駆除方法のわからない新種のウイルスを解析することが可能である。このシステムの検証として３つのウイルス（Blaster，Nimda，Sasser）をテスト解析し、順次攻撃、ローカル攻撃、ランダム攻撃の３つの感染パターンを抽出できた。

**キーワード**：ウイルス，ワーム，視覚化，感染パターン，Blaster，Nimda，Sasser，仮想環境

## Virus-Worm Infection Pattern Analysis

Yoshiyuki SEINO and Ichirou MORIGUCHI

We constructed a system that captures access patterns and infection speeds of virus-worms in a virtualized environment, and that analyzes and visualizes them. This analysis system plots the correlation graphs of virus's target IP addresses, and visualizes the change of target addresses on movies. Futhermore, the analysis system can analyze unknown viruses to which no one knows removal methods by using of a virtualized environment. Three well-known viruses have been used for the check of this system, and it becomes clear that this system can extract three infection patterns, i.e., random-attack, sequential-attack, local-attack.

**Keyword**：Virus, Worm, Visualization, Infection Pattern, Blaster, Nimda, Sasser, Virtualized Environment

*東京情報大学総合情報学部 情報システム学科
Tokyo University of Information Sciences, Faculty of Informatics, Department of Information Systems
**東京情報大学総合情報学部 情報システム学科
Tokyo University of Information Sciences, Faculty of Informatics, Department of Information Systems

2008年11月27日受理

## 1．はじめに

今やインターネットはインフラストラクチャーとして必要不可欠な存在となっているので、インターネットを介して行われるウイルス・ワーム（以後ウイルスとまとめる）の感染は、私たちの社会を脅かす危険な存在である。そのため、ウイルス対策のための調査研究は現在も活発に行われているが、それらはウイルスシグニチャを使ったウイルス検知手法がほとんどである。ウイルスは感染を拡大させるために様々なIPアドレスに大量のアクセスを行うが、そのアクセス速度やアクセスパターンは十分に調査されていない。このようなウイルス感染パターンを解析することは、今後のウイルス感染対策のために重要であると考えられる。

ウイルス感染の解析には感染活動を記録したログファイルを用いるが、ログファイルの中には大量の情報があり、その一つ一つを目で追っていくのは非効率的である。そこで、ログファイルの内容を視覚化し、より理解しやすい情報の抽出が必要であると考えられる。

本研究では仮想ネットワーク環境下でウイルスに実際に感染活動を行わせ、その感染速度やアクセスパターンを収集し、解析・視覚化を行うシステムの構築を行った。この解析システムでは、感染パターンを独自の手法で表現し、感染速度を考慮して動画で表示することとした。また、ウイルスの感染先IPアドレスの変化に相関が存在するかを解析するために、プロット図も表示可能にした。最後に、システムを検証するためによく知られている3種のウイルスをテスト解析し、3つの感染パターンを抽出することができた。

## 2．解析システム

### 2.1　解析システムの概要

実際にウイルスを実行するにあたって、研究環境外へウイルス感染が拡大しないよう閉じた仮想ネットワークを構築し、その内部で感染活動の解析を行った。

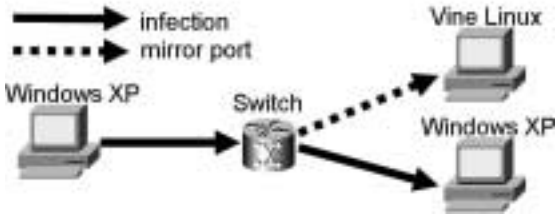


**図1．解析システム概要図**

仮想ネットワーク内には、ウイルス感染したもの、ウイルスに感染する可能性のあるもの、感染活動を監視するもの、合計3台のコンピュータを用意した。これら3台のコンピュータにはローカルIPアドレスが割り振られ、相互に通信が可能となっている。

今回テスト解析を行うウイルスとしては、ウイルスの多くから標的とされるWindowsへ感染するものを対象とすることにした。そのため、ウイルス感染したコンピュータとウイルス感染の可能性のあるコンピュータの2台のOSにはWindows XPを使用した。感染活動を監視するコンピュータのOSとしては、今回使用するウイルスからの感染を受ける心配がなく、OSとしてのサイズが大きすぎず、かつ開発環境として必要十分な機能を持っているVine Linux 4.2を採用した。

開発した解析システムはパケットキャプチャから視覚化までが自動化されているので、使用者が本システムを利用して解析を行うには、ウイルスの実行と解析システムのシェルスクリプトを起動するだけでよい。

### 2.2　仮想ネットワーク

仮想ネットワークの構築にはVMwareServerを利用した。VMwareServerはホストOS上に複数のOSをゲストOSとしてインストールでき、それらを相互に通信可能とするソフトウェアである。また、VMwareServerはゲストOSとして使用するWindowsとLinuxの動作を保障している他、無料で利用可能で、さらに、スナ

ップショットという機能を利用できる。スナップショットはゲストOSの状態を一時的に保存しておき、ゲストOSを任意のタイミングで保存した状態に戻す事ができる機能である。これにより、感染を受けたゲストOSを以前の未感染の状態に容易に戻す事ができ、複数回のウイルスの解析を効率よく行うことができる。

### 2.3 キャプチャ方法

感染活動のキャプチャには、CUIベースのethereal であるtetherealを監視役のVine Linux上で使用して行った。

tetherealでキャプチャした結果は通常コンソールに表示されるので、これをファイルへ保存する方法としてはリダイレクションが考えられるが、それでは全てのパケットを保存できないことがある。これは、Linuxがtetherealからの出力を逐一受け取り、その度に標準出力としてファイルに書き込むという手間があるためである。この問題を回避するために、tethereal自身でファイルに書き込みを行うように-wオプションを使用した。以下は-wオプションの使用例である。

例：tethereal −w output.log

このオプションを使用して作成したファイルはlibpcap方式で保存されてしまうが［1］、通常のテキスト形式で保存した方が後の処理が容易なので、キャプチャ作業後、tetherealに-rオプションでファイルを読み込ませてリダイレクションを行い、テキスト形式のログファイルを作成することとした。以下はそのコマンド例である。

例：tethereal −r output.log > output.txt

## 3．感染パターン視覚化

### 3.1 手法

感染パターンの視覚化には感染活動のあて先となったIPアドレスを基に、感染速度を考慮して動画表示で行う。この際、IPアドレスは$2^{32}$個まで表現できるため、IPアドレスと画面の点1つを対応させて表示しようとすると、現在使用されている画面の解像度をはるかに超えた巨大な解像度が必要になってしまう。そのため、今回の研究では表示するIPアドレスをオクテットごとに分解し、分解した4つの値をそれぞれに対応した枠のX座標位置に縦線で示し、合計4本の縦線で1つのIPアドレスを表現することとした。


**図2．視覚化プログラム実行図**

### 3.2 動画プログラム

動画表示には、Java言語のAWT機能を利用した［2］。AWTとはJava言語に標準で用意されているグラフィック関連のライブラリである。このAWTに含まれているクラスを利用することにより、ウィンドウの表示、文字や線などの表示を行える。今回の感染パターンの視覚化には、ウィンドウを表示し、そのウィンドウに描写を行って実現した。

また、感染活動を動画のように見せるためには定期的にウィンドウ内部の描写を行い、描写内容を刻々と変える必要がある。これにはjava.util.TimerTaskとjava.util.Timerをimportし、描写処理を定期的に実行させるクラスを作成することで実現した。具体的には、TimerTaskクラスを継承したMyTimerTaskクラスのrunメソッドに描写処理などを記述し、Timerクラスのscheduleメソッドにその MyTimerTaskクラスを登録することで定期的な描写を行っている。

動画表示のための定期的な描写処理に際し、注意すべき点がある。描写内容を変化させて動

きを見せるためには、再描写のはじめに一度画面を初期化した後、新たな描写を行うことになる。この再描写の一連処理をウィンドウ上に直接行ってしまうと、初期化と新たな描写が全く同時に行われるわけではないために、表示が点滅しながら変化していくように見えてしまう。以下はその描写のプログラム例である。

```
paint（Graphics g）{
  g.setColor（Color.BLUE）；
  g.fillRect（0,0,640,480）；//初期化処理
  g.drawImage（smileImage,x,y,this）；//描写処理
  x = x + 100；y = y + 100；
}
```

上のプログラム例を実行すると、図3のように処理が進み、表示画面に対して初期化と描写が交互に行われるため、描写されたものが一度消えてから出てくるので点滅しているように見えるのである。

この問題を起こさないために、ダブルバッファリングという手法を用いた。この手法は、実際に表示するウィンドウ以外にもう一枚表示されない仮想的なウィンドウを用意し、全ての再描写処理を仮想的なウィンドウに行ってから、仮想的なウィンドウの内容を表示されているウィンドウに一気に書き込む方法である。これにより、表示画面は初期化の処理の影響を受けずにすみ、表示が点滅することはなくなる。以下はそのプログラム例である。

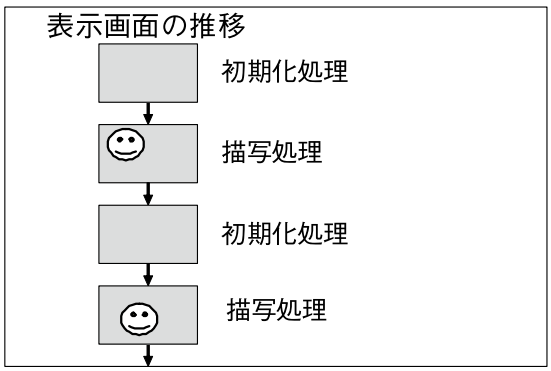


**図3．画像の点滅が発生する画面描写方法**

ダブルバッファリング

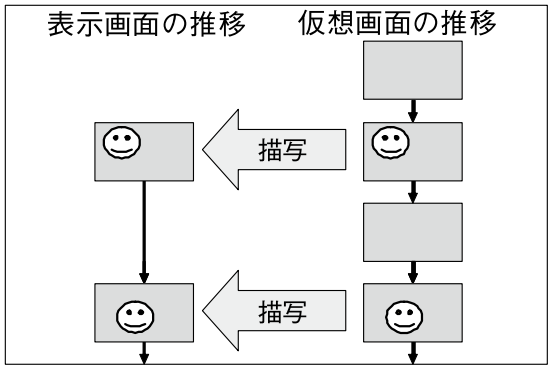


**図4．ダブルバッファリングを用いた画面描写**

```
paint（Graphics g）{
  Image offImage = createImage
  （640,480）；//仮想画面
  Graphics og = offImage.getGraphics（）；
  og.setColor（Color.BLUE）；
  og.fillRect（0,0,640,480）；//仮想画面初期化
  og.drawImage（smileImage,x,y,offImage）；
  //仮想画面に描写
  x = x + 100；y = y + 100；
  g.drawImage（offImage,0,0,this）；//仮想画面を表示画面に描写
}
```

上のプログラム例を実行すると、図4のように処理が進み、表示画面は点滅せずに動画表示を行うことが可能となる。

### 3.3　動画表示

感染速度を考慮して表示を行うために、動画表示プログラムがログファイルから感染活動の時間を抜き出して利用し、感染パターンの表示タイミングを調整した。その際、感染パターンを表示し続けると、後から表示される感染パターンがどのように表示されているのかわからなくなってしまう。そこで、表示した感染パターンは時間の経過と共に減色させ、最終的には白い跡をアクセス痕として残すことにした。

また、ウイルスの感染活動が非常に速いため、感染活動の時間を現実の時間に合わせて表示を

行うと、複数の感染パターンが同時に表示されてしまう。そのため、動画表示は実際の感染速度の100倍の時間でゆっくり表示することとした。

ウイルスによっては、感染活動中にしばらく何もしない時間があるので、感染活動の時間相関を解析する場合を除き、このような待ち時間を見つけた場合にはソフトウェア内部の時間を進め、次の感染活動をすぐに表示することとした。

### 3.4 その他補助情報

感染パターン解析に有用なその他の情報は、ウィンドウ上で補助的に表示することとした。

動画の表示時間は実際の時間と異なるため、動画のウィンドウ左上部に常に表示することとした。また、感染活動が全くない時間を自動で進めた場合にも、時間を進めたことをウィンドウ左上部に表示することとした。

感染パターンを表示する際に、感染活動が時系列に沿ってどのような頻度で来ているかをウィンドウ下部に表示することとした。ウィンドウ下部に濃い灰色で四角く塗りつぶしたフレームを準備し、感染が行われた時に赤い縦線をフレーム内右端に描写する。赤い縦線は時間に応じて左へと移動し、時系列表示用フレームの左端に達すると描写されなくなる。

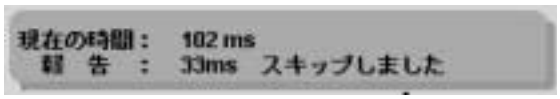


**図5．時間表示部分**

**図6．時系列表示部分**
右から左へと感染活動を示す縦棒が流れていく

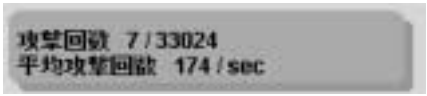


**図7．攻撃回数表示部分**

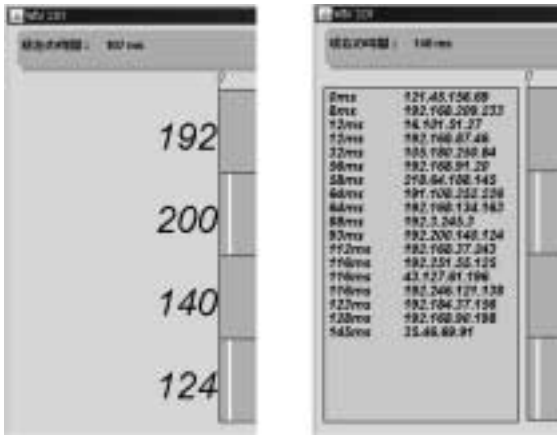


**図8．左図が数値表示、右図がログ表示**

ウイルスの感染活動の速度を理解するために、秒間平均攻撃回数と現在までに視覚化を行った攻撃回数をウィンドウ上部に表示した。

表示した感染パターンのIPアドレスの値を、数値またはログ形式で表示可能とした。数値表示では各オクテットの値をそれぞれ対応する枠の左に表示し、ログ表示ではウィンドウ左にログ表示用の枠を設けて枠内に記述した。

## 4 ウイルス・ワーム

### 4.1 ウイルスの概要

今回構築した解析システムの検証のためにテスト解析を行ったウイルスは、Blaster、Nimda、Sasser の 3 種である。

Blasterは2003年 8 月に多数のWindowsに感染したウイルスである。コンピュータをインターネットに接続しているだけで感染するため、全世界に蔓延した。このウイルスは、ソフトウェアコンポーネント同士がネットワークを介して通信することを可能とする分散オブジェクト技術の１つであるRPC DCOMの脆弱性を利用して135番ポートへ感染を行う［３］。Windows XPの初期設定ではこのRPC DCOMの機能を使用するようになっており、サーバだけでなくクライアントでも起動されている

［4］。

Nimdaは2001年9月に多数のWindowsに感染したウイルスである。このウイルスは添付メールによる感染や、ホームページアクセス者への感染、Microsoft IISと呼ばれるWebサーバへの感染など多数の感染経路を持っている。今回の研究では感染先のIPアドレスに着目して視覚化を行うので、Microsoft IISの脆弱性を利用した感染方法に限定して解析することとした［5］。

Sasserは2004年5月に多数のWindowsに感染したウイルスである。前述のBlasterと同じく、インターネットに接続されたコンピュータへ感染を行い全世界に蔓延した。このウイルスはログオンに関する処理を管理するLSASSの脆弱性を利用して他のコンピュータへと感染を行う［6］。

### 4.2　解析結果

今回の解析では、攻撃先IPアドレスの値の変化を調べるために、連続して送られた攻撃先IPアドレスの第4オクテットの値2つを組とし、前の値をX座標として、後の値をY座標としてプロットさせた。このように値変化をプロットすることで、攻撃先の選定方法に相関が存在するかを調べた。

#### 4.2.1　Blaster

Blasterは、感染先を決める方法を2つ持っていることがわかった。1つは、IPアドレスの上位3オクテットをランダムな値としたA.B.C.Xについて、第4オクテットXを順次1から1つずつ増加させて攻撃していく方法。もう1つは、IPアドレスの上位2オクテットに感染元コンピュータのローカルIPアドレスの上位2オクテットを使用し第3オクテットにはランダムな値を使用したE.F.G.Xについて、第4オクテットXを順次1から1つずつ増加させて攻撃していく方法である。2つの方法のうちどちらをどれほどの確率で選ぶか調べるためにBlasterを100回解析したところ、前者を47回、後者を44回選び、残りの9回は全く攻撃をしていなかった。このウイルスについてはソースコードを入手出来たのでソースコードを解読したところ、このウイルスは前者のランダムな値を使用した際に、コンピュータがIPアドレスとして使用できない第1オクテットの値が224以上のIPアドレスを選んでしまうことがあることがわかった。その際の動作を調べるためBlasterのソースコードを無害化して、実験的に第1オクテットの値を224以上としたIPアドレスに対し、元のソースコード同様にconnect関数［7］を利用して接続を行うようにしたところ、パケットが送信されないことが判明した。これは、100回解析した際にBlasterが感染活動を行わなかった9回がそれに当たると考えられる。また、ソースコードを解読した結果、前者のランダムな値を使用する方法を60%、後者の方法を40%で使用するようにコーディングされていた。100回の解析結果における前者の攻撃方法を行った47回と感染活動を行わなかった9回を足した56回と、後者の攻撃方法を行った44回とを実際のソースコードの確率と比較すると多少のズレは存在するものの、ほぼ正しい解析結果が得られていることが確認できた。

このように、Blasterは2種類の方法を使うことによって、感染したコンピュータから物理的に近いローカルエリアネットワークに対して感染を拡げることと、全く別のネットワークへと攻撃を拡大させることを両立しているものと推測される。

図10は、Blasterの攻撃先IPアドレスの第4オクテットの値変化をプロットしたものである。なお、この図の作成にはgnuplotを用いた。この図を見ると、BlasterがIPアドレスの値を順次に攻撃していることがわかる。

その他の特徴として、秒間平均攻撃回数がおよそ11回であったことと、1度の攻撃で20個のあて先に向けて攻撃を行った後およそ1.8秒後に再度攻撃を行うことがわかった。この1.8秒の待ち時間は、20個まとめて送信を行った後、それらの通信を終了させるために設けているも

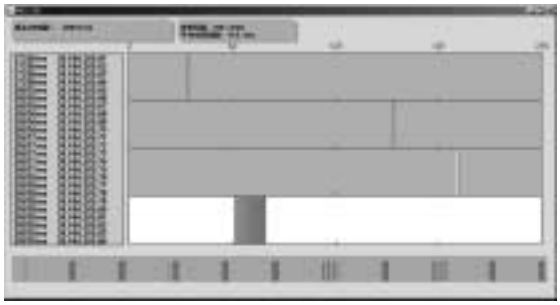

**図9．Blasterの感染パターンを動画表示している様子**
第４オクテットを１から順に攻撃している

**図11．Nimdaの感染パターンを動画表示している様子**

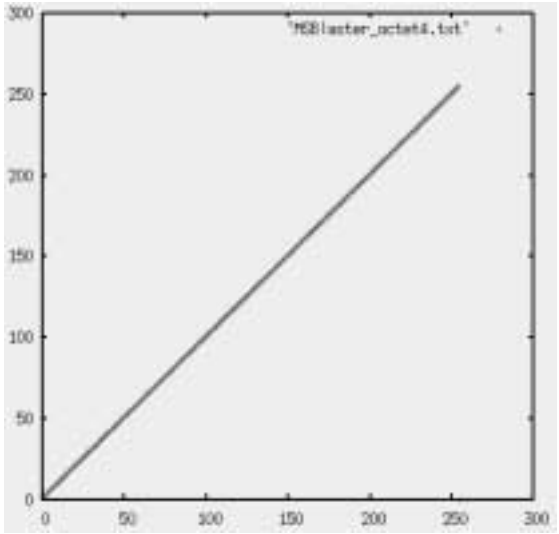


**図10．Blasterの攻撃先IPアドレスの第4オクテットの値変化をプロットした図**

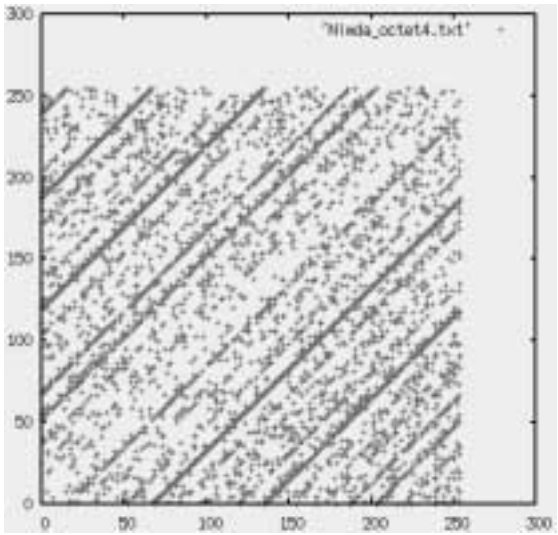


**図12．Nimdaの攻撃先IPアドレスの第4オクテットの値変化をプロットした図**

のと推測できる。実際、地球上の物理的に最も遠いホストに対するパケットでもRTTは300ms程度であり、遅延が発生した場合でも1300ms程度であるから、1.8秒の待ち時間を設ければ通信を開始するのに十分であることがわかる。

### 4.2.2 Nimda

Nimdaは攻撃パケットの送り方を複数行っていることがわかった。１つ目は、感染元コンピュータのIPアドレスの第１オクテットを攻撃先の第１オクテットに固定して、残りの第２第３第４オクテットを擬似的な乱数値としたIPアドレスへと攻撃する方法。２つ目は、感染元コンピュータのIPアドレスの第１第２オクテットを攻撃先に使用し、残りの第３第４オクテットを擬似的な乱数の値としたIPアドレスへと攻撃する方法。３つ目は、全てのオクテットを擬似的な乱数としたIPアドレスに攻撃をする方法である。

擬似的な乱数値とは、独自の計算でランダムなように数値を動かしているだけで、実際は何

らかの規則性を持った値が入れられている。

図12は、Nimdaの攻撃先IPアドレスの第4オクテットの値変化をプロットしたものである。多少ランダムに見える部分も存在するが、右上がりの直線が複数本見える。また、これは第1第2第3オクテットの変化についても同様の結果が得られた。これは、このウイルスの値の決め方が独自の計算式に基づいていることを示している。

その他の特徴として、秒間平均攻撃回数はおよそ24回ということがわかったが、このウイルスは同じIPアドレスへ2度攻撃する特徴があったため、実際には秒間12個のIPアドレスへ向けて攻撃していることになる。

### 4.2.3　Sasser

SasserはNimda同様に攻撃パケットの送り方を複数行っていた。1つ目は、感染元コンピュータのIPアドレスの第1オクテットを攻撃先の第1オクテットに固定し、その他のオクテットの値をランダムに変えて攻撃する方法。2つ目は感染元コンピュータのIPアドレスの第1第2オクテットを攻撃先に使用し、残りの第3第4オクテットをランダムに変えて攻撃する方法。3つ目は、完全にランダムなIPアドレスへと攻撃する方法である。これら3つを同時に行って、感染を拡大させようとしていた。

図14は、Sasserの攻撃先IPアドレスの第4オクテットの値変化をプロットした図である。この図を見た限り、前後の攻撃の送り方には相関

**図13．Sasserの感染パターンを動画表示している様子**

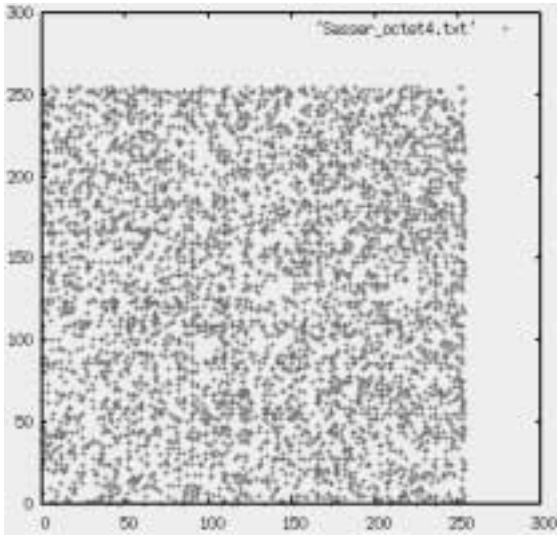


**図14．Sasserの攻撃先IPアドレスの第4オクテットの値変化をプロットした図**

関係が見られない。また、第1第2第3オクテットの値もプロットした結果は、第4オクテットと同様の結果が得られた。

その他の特徴として、秒間平均攻撃回数はおよそ18回であったが、このウイルスは同じIPアドレスへ3回攻撃を行っていたため、実際には秒間6個のIPアドレスへと攻撃を行っていることになる。

### 4.3　感染パターンの分類

4.2の結果から、ウイルスには感染パターンが複数存在することがわかった。このことから、ウイルスの攻撃の種類分けを行った。

### 4.3.1　順次攻撃

これは攻撃対象のIPアドレスの値を1つずつ増加させて順次に総当り攻撃していく手法である。この手法は、特定のネットワーク単位に向けて攻撃を行う時などに用いられるものと考えられる。ネットワークでのIPアドレスの割り振り方には通例として1から1つずつ増加させた値を使用する傾向があるため、なるべく早く感染を拡大させるためには1から1つずつ攻撃を行うのが効率的だからである。

### 4.3.2　ランダム攻撃

これは攻撃対象のIPアドレスの値をランダムに変化させて攻撃していく手法である。全世界のコンピュータへ感染を行わせようとする時には、異なるネットワークのIPアドレスへと攻撃を行わなくてはならないため、このような手法が存在するものと考えられる。また、サーバは内部ネットワークより外部ネットワークに多く存在するため、この手法は特にサーバ系に感染する場合に有効だと考えられる。

### 4.3.3 ローカル攻撃

これは感染しているPCと同じIPアドレスの上位オクテットを攻撃先IPアドレスに使用し、ローカルネットワーク内部へと攻撃を行う手法である。ローカルネットワーク内部には複数のクライアントコンピュータが接続されていることが多く、また、同一ネットワーク内のコンピュータ同士では通信が非常に小さなRTT値で行われるため、この手法は素早く多くのクライアントへ感染を拡大させるために用いるものと考えられる。

## 5．まとめ

本研究では、ウイルス・ワーム感染パターン解析システムの構築と、実際にいくつかのウイルスに対し感染パターン解析を行った。この解析システムは仮想環境を利用しているため、未知のウイルスの解析も可能であり、未知のウイルスの感染対策につながる情報の抽出が可能である。さらに、独自の手法で感染パターンを視覚化するとともに、ウイルスの攻撃先の値変化に相関が存在しないかをプロット図で表示することができる。また、いくつかのウイルスを使ったテスト解析により、このシステムがウイルスの種類ごとの特徴的な感染パターンを抽出できることを実証した。

今後、本研究のような詳細な感染パターン調査を行うことによって、感染に強いネットワーク構成、感染の抑止方法、感染を遅らせる手法の開発につながると考えられる。

## 【参考文献】


[1] Etherealを使おう，Ethereal_保存ファイル名を指定する（Tethereal） at http://www.space-peace.com/ethereal/ethereal_11_3.htm（2003）.

[2] Developer Resources for Java Technology, Java 2 Platform SE 5.0 at http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/（2004）.

[3] Symantec - AntiVirus, Anti-Spyware, Endpoint Security, Backup, Storage Solutions, W32.Blaster.Worm at http://www.symantec.com/region/jp/sarcj/data/w/w32.blaster.worm.html（26 Feb. 2004）.

[4] Geekなページ［インターネット技術メモ］, Geekなページ：DCOM（分散COM）を無効にする at http://www.geekpage.jp/practical/winxp-tips/dcomcnfg.php（2008）.

[5] Symantec - AntiVirus, Anti-Spyware, Endpoint Security, Backup, Storage Solutions, W32.Nimda.A@mm at http://www.symantec.com/region/jp/sarcj/data/w/w32.nimda.a@mm.html（15 Jan. 2003）.

[6] Symantec - AntiVirus, Anti-Spyware, Endpoint Security, Backup, Storage Solutions, W32.Sasser.Worm at http://www.symantec.com/region/jp/sarcj/data/w/w32.sasser.worm.html（6 May. 2004）.

[7] JM Project , Manpage of CONNECT at http://www.linux.or.jp/JM/html/LDP_man-pages/man2/connect.2.html（11 Oct. 2008）.