平成 16 年度

修　士　論　文

題　目　　IDS を用いたモバイルエージェントによる
セキュリティパケットフィルタリングの実装と評価

氏名　　長田　昌訓

平成 15 年度入学

島根大学大学院　総合理工学研究科博士前期課程

数理・情報システム学専攻　計算機科学講座

指導教員　　田中　章司郎　教授

平成 17 年 2 月 21 日受理

目次

第１章　序論

近年，ブロードバンドが普及し，それに伴い不正アクセスによる攻撃や情報の漏洩などネットワークセキュリティが深刻な問題となっている．

外部からの不正アクセスを防ぐために，ゲートウェイにファイアウォールを設置する方法が使用される．しかしハッカーの技術的な向上により，外部からの攻撃であってもファイアウォールを通過して内部に浸透し，そこからウァーム等の新たな脅威となる可能性がある．ゲートウェイ上のファイアウォールは通常 LAN 内部からの攻撃（インサイダー攻撃）ではなく外部攻撃から LAN を保護する．したがってインサイダー攻撃の場合，LAN 上のホストを脅かすことになる．その防衛方法として，パーソナルファイアーウォールの導入がある．しかし市販のパーソナルファイアウォールなどは主要なポート以外を常時ブロックするため，インターネットを使用するアプリケーションツールで必要なポートが閉じて利用できないといった問題がある．

また，別の方法として進入検知システム（Intrusion Detection System ：以下 IDS）よる監視といった方法があるが，不正アクセス検知時にネットワーク管理者が不在である場合，迅速な対応がとれずにセキュリティポリシーの変更まで攻撃されるといった問題がある．

本研究では，上記の問題を解決するために，モバイルエージェントを用いることで，IDS の機能と連動してリアルタイムでネットワークに対する不正アクセスを監視し，ポートを常時ブロックするのではなく，不正アクセスが行われた場合に攻撃パケットの発信元 IP と被害先 IP を特定し，ネットワーク全体を守るため各マシンに自動的に不正にアクセスをしようとした IP アドレスを弾くパケットフィルタを起動させるモバイルエージェントを用いたセキュリティシステムを試作し，検証を行った．

第 2 章　先行研究との比較

本章では，既存のセキュリティシステムについて述べ，本研究との比較を行う．

ホストベースのセキュリティとして，以下の項目が挙げられる．

（1）パーソナルファイアウォール

（2）IDS のみ

（3）IDS＋モバイルエージェント＋パーソナルファイアウォール

（1）の方法は個々のマシンに導入する場合の手間とコストがかかり，パーソナルファイアウォールは主要なポート（21，80 など）以外を常時ブロックするため，インターネットツールで利用ポートが閉じていると利用できないといったデメリットがある．

（2）の方法は不正アクセスの検知はできるが，原則防御することができない．IDS とともにゲートウェイファイアウォールを用いた方法がある．

（3）は，通常開いているポートでもブロック可能であり，IP アドレスによるフィルタにより未知の攻撃にも対応できる．また導入してからは管理者以外の専門知識がいらないといった利点がある．本研究は（3）を用いる．

先行研究の調査として以下の 2 つのサイトで検索を行った．

●INSPEC にて以下のキーワードを元に検索（検索日：2003 年 8 月 9 日）

Security & Mobile agent	586 件　—①
Security & aglet & Firewall	0 件
Security & Mobile agent & aglet	15 件　—②
Security & Mobile agent & Firewall	14 件　—③
Security & Mobile agent & IDS	11 件　—④
Packet filter & Mobile agent	5 件　—⑤
Packet filter & aglet	0 件

INSPEC とは英国電気学会（IEE）が提供する電気・電子・制御・情報工学関連の英文二次文献データベースで世界中の理工系の雑誌の文献がこのデータベースに登録されている．

①では該当件数が多かったため，②③④とキーワードを追加したのち，⑤を含めて調査を行った．上記（3）に属するエージェントを用いたセキュリティシステムの先行研究として，ネットワーク上に分配されたファイアウォールが個々でホストにアクセスコントロールを課す研究[1][2]があるが,これは各マシンに IDS を必要とする．これに対し

て本研究では IDS はモニタマシン１台にのみ導入すればよい.

● 特許庁特許電子図書館にて以下のキーワードを元に特許・実用新案の検索（検索日：2003 年 8 月 9 日）

モバイルエージェント & ファイア & ウォール	０件	
モバイルエージェント & ファイヤ & ウォール	０件	
モバイルエージェント & セキュリティ	5 件	—⑥
Mobile & Agent & Security	０件	
モバイルエージェント & IDS	０件	
モバイルエージェント & パケットフィルタ	０件	

特許の事例としては⑥のキーワードで５件が該当し調査した.
その中で, 分散ネットワークコンピューティングシステム（特許公開番号 2003-022189）が先行研究として挙げられたが, 扱う範囲が広く具体的な実装については記載されていなかった.

第3章　IDS

3.1　IDSとは

侵入検知システム（Intrusion Detection System）と呼ばれ，コンピュータやネットワークに対する不正アクセスなどを検出し，システム管理者などに知らせてくれる警報装置を指す．中でも特定のコンピュータのみを監視するIDSを「ホストベースIDS（HIDS)」，ネットワークを監視するIDSを「ネットワークIDS（NIDS)」と呼び区別される[3].

IDSを用いる目的としては不正アクセスに対して被害状況の記録，被害の防止および軽減，侵入者の特定，セキュリティの評価などが挙げられる．

3.2　snortの導入

本システムではIDSにsnort-2.0.0（http://www.snort.org/）を用いた．なお2004年2月10日現在の最新バージョンはSnort2.3.0となっている．

3.2.1　snortとは

Snortは Marty Roesch氏を中心にして開発され，GNU General Public Licenseのもとでオープンソースとして公開されているNIDSである．Snortは NIDSをはじめとして，パケットスニファ，パケットロガーとしての機能を持ち，入力パケットを解析し，予め用意されているシグネチャ（攻撃方法のパターン）とパターンマッチングを行いシステムへの侵入検知を行う（図 3.1）．Linux を始めとして多くのプラットホームで稼動可能である．

Snortはシグネチャベースの IDSで，ネットワーク上を流れるパケットを検知するためのルールを持っている．ルールにはアラートを発生させるためのトリガーとなる条件，つまりシグネチャが記述されている．もしこのパケットがシグネチャにマッチしたのなら，ルールはアラートを発生させ，ログファイルやデータベースに対してそのアラートを送ることが可能である[4].

Snortの解説，導入については以下のサイトを参照していただきたい．

snortの導入：http://jem.serveftp.com/

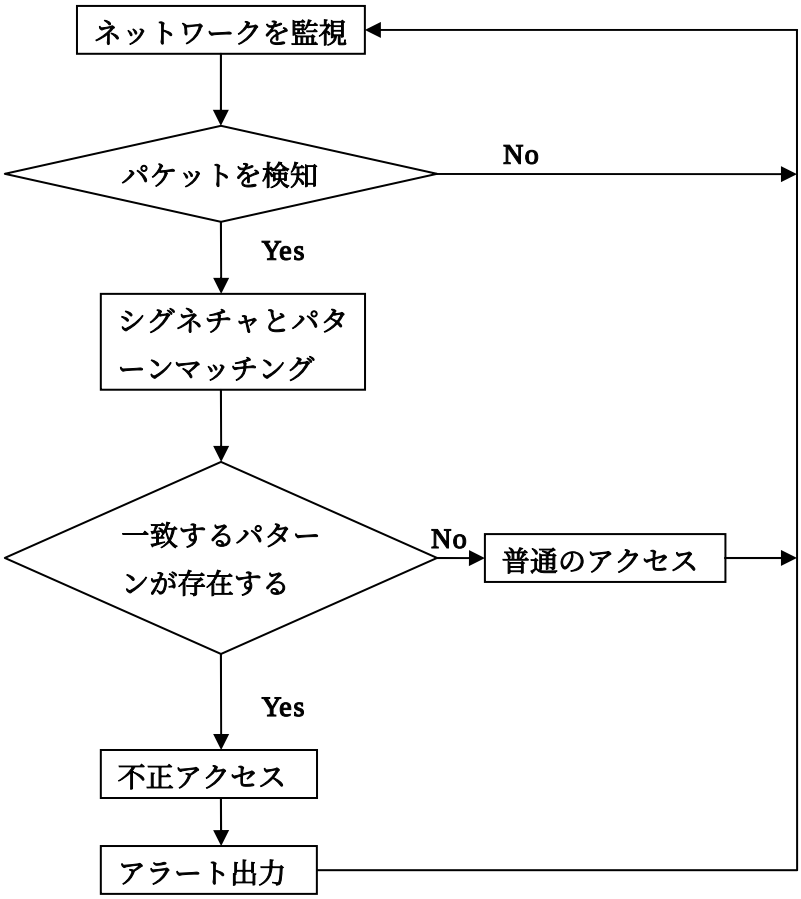

図 3.1　snort の不正アクセス検知の流れ

3.2.2　パケットキャプチャドライバ

本システムは Windows 版の Snort の snort を用いたため，稼動するには Windows Packet Capture library（WinPcap）というパケットキャプチャドライバをインストールする必要がある．以下の URL より入手できる．

Windows Packet Capture Library：http://netgroup-serv.polito.it/winpcap/

また，Linux で Snort が動作するためには「libpcap」がインストールされていることが必要．これは LAN ケーブル上のパケットをモニタリングする UNIX システムにおいて使用可能な一般的なライブラリで，以下の URL で入手できる．

TCPDUMP public repository：http://www.tcpdump.org/release/

第 4 章　エージェント

4.1　エージェントとは

エージェントは代理人と訳され, コンピュータ用語としても同じような役割を持っており，ユーザあるいはプログラムからの指示でなく，与えられたルールを基にして場面に応じて自立的に自分自身の動作の決定を行うようなソフトウェアシステムを指す.

特にあるコンピュータから別のコンピュータに移動させて実行することの可能なエージェントをモバイルエージェントと呼ぶ.

4.2　ASDK

本システムでは Java を用いてモバイルエージェントを作成しているが，そのプログラム環境として IBM 東京基礎研究所で開発され，現在はオープンソース化され SourceForge.net（http://sourceforge.net/projects/aglets/）で公開されている Aglets Software Development Kit 2.0.2（ASDK 2.0.2）を用いた.

ASDK を使用した理由としては，他のモバイルエージェント構築ツールに比べ比較的新しかった点，java がベースなので OS に依存されにくい点，オープンソースであった点が挙げられる.

4.2.1　ASDK のインストール

SourceForge.net よりダウンロードしたファイルを解凍後, bin, conf, lib, META-INF, public の 5 つのフォルダと INSTALL.html，LICENSE.html の 2 つの html ドキュメントが作成される．bin フォルダ内にインストール用のバッチファイルが用意されているので，ant.bat を実行し，コマンドプロンプトにて ant install-home をタイプする．Ant.bat 実行時に READ ME が表示されるので詳細はそちらを参照していただきたい.

4.2.2　ASDK の環境設定

ASDK を使用するためには以下のように環境変数を設定する必要がある.

・ ディレクトリの構成（例）

Aglets 2.0.2	C:¥aglets-2.0.2

・ 環境変数

AGLET_HOME	C:¥aglets-2.0.2
AGLET_PATH	C:¥aglets-2.0.2¥public
AGLET_EXPORT_PATH	C:¥aglets-2.0.2¥public

Path	C:¥j2sdk1.4.1_02¥bin;%AGLET_HOME%¥bin
CLASSPATH	%AGLET_HOME%¥lib¥aglets-2.0.2.jar

4.2.3 Aglet サーバ

ASDK において移動エージェントは Aglets とよばれる．Aglets はインターネット上のひとつのホストから別のホストへ移動することのできる Java のオブジェクトである．Java 環境には Java 2 SDK 1.4.2_03 を用いた. 複数のマシン上で Aglet を動かす場合, 事前に各計算機上で Aglets サーバと呼ばれる Aglets を管理するプログラムを稼動させる必要がある．そのためモニタするマシン全てにあらかじめ Aglets サーバを立てておく必要がある．

Aglet サーバは基本的に GUI を持たないが，Tahiti と呼ばれるプログラムを利用することで Aglets サーバを GUI が管理することができる. Tahiti を起動すると自動的に Aglets サーバが起動されるので, GUI を装備した Aglet サーバを動作させるには Tahiti を起動すれば良い[5].

4.3 Aglet プログラム

ここでは簡単な例を用いて Aglet プログラムと Tahiti の使用法を説明する．

4.3.1 **プログラム例**

プログラムファイル（表 4.1）：ExampleAglet.java
内容：マシン amagi からマシン osada へ移動する Aglet プログラム

表 4.1　Aglet プログラム例

```
1: import com.ibm.aglet.*;
2: import java.net.*;
3:
4: public class ExampleAglet extends Aglet{
5:     public void onCreation(Object init){
6:
7:         // osada へ移動する
8:         try{
9:             URL destination = new URL("atp://osada:4434");
10:        } catch(MalformedURLException mue){
11:          System.out.println(mue);
12:        }
13:        try{
```

```
14:          dispatch(destination);
15:      } catch(Exception e) {
16:              System.out.println(e.getMessage());
17:      }
18:  }
19:  public void run(){
20:      System.out.println("Hello");
21:  }
22:}
```

表 4.1 の 1 行目では Aglet 作成に必要なパッケージ com.ibm.aglet.*を import している．4 行目で示すようにすべての Aglet は Aglet クラスを継承する必要がある．本例では Aglet クラスが持つメソッドとして onCreation メソッド，run メソッドを使用している．5 行目の onCreation は生成されるときに呼ばれるメソッドであり，呼ばれると amagi 上の Aglet サーバに送信（dispatch）される．Aglet 送信は 14 行目のように dispatch メソッドを呼び出すことで実行される．Dispatch メソッドは URL を引数として，その URL に Aglet を移動させる．本例では 9 行目で URL は「atp://osada:4434」と指定されている．

4.3.2 Tahiti の使用例

● Tahiti 起動コマンド

```
C:¥aglets-2.0.2¥bin> agletsd -f ..¥cnf¥aglets.props
```

① Tahiti を起動すると図 4.1 のような画面が現れる．

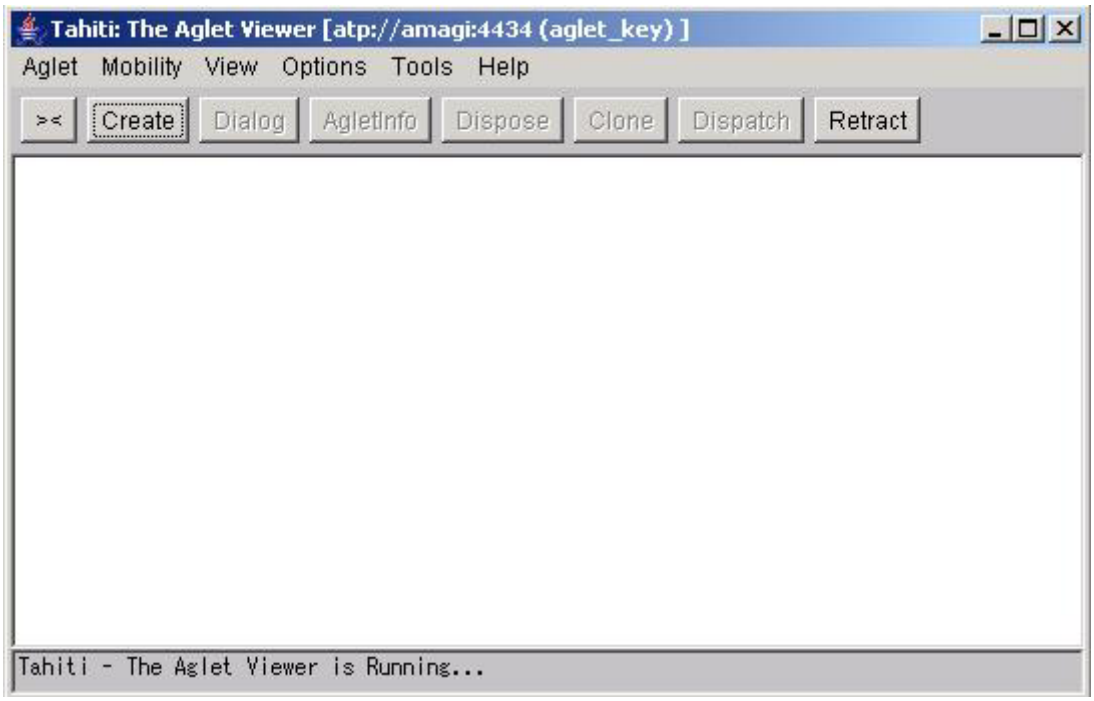

図 4.1　Tahiti 起動画面

「atp」とは agent transfer protocol の略称で，Aglet の移動を定めたプロトコルである．amagi はマシン名であり，Tahiti が使用するポート番号は TCP4434（デフォルト値）である．

② プログラムを javac でコンパイルし，生成されたクラスファイルを環境変数 AGLET_PATH で指定したディレクトリに保存する．Create Aglet ウインドウの Aglet name に名前を入力し Aglet のクラスファイル名を入力し「Create」を押す．（図 4.2）

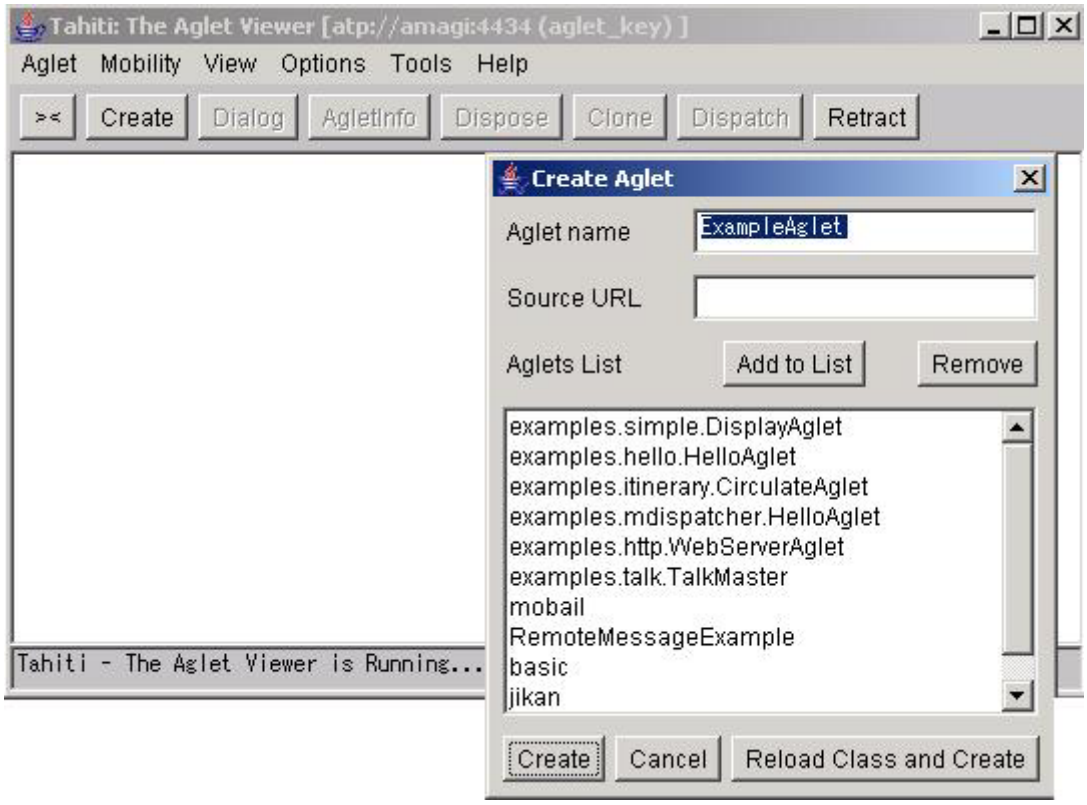

図 4.2　Aglet 生成

なお Tahiti の詳細な動作説明については以下を参照.

Tahiti User's Guide：http://www.trl.ibm.com/aglets/tahiti1.1/index_e.htm

第 5 章　システム概要

本システムを述べるにあたって以下の 4 点が前提であるとする. これらはエージェント自身のセキュリティ問題となるため，本研究と意味合いが異なるためである.

（1）各マシンで Aglet サーバは常に起動している状態であり，エージェントの移動を妨げることはない.
（2）本システムの実行を行ったシステム管理者以外は，その内容を知ることができない.
（3）管理者は悪意を持つことはなく，エージェントを悪用しない.
（4）モニタマシンと被害先マシンの通信は安全であるとし，第三者から攻撃を受けることはない.

5.1　構成要素

本システムの構成要素と名称を表 5.1 にまとめる.

表 5.1　システムの構成要素

名称	内容
システム管理者	本システムに対して責任を負う利用者.
モニタマシン	基本エージェントの実行環境かつ IDS の実行環境であり，外部から被害先マシンへのパケットの流れを監視可能な環境.
被害先マシン	システム管理者が管理し，移動エージェントの実行環境. 複数台存在する.
攻撃元マシン	被害先に対して IDS が検知する攻撃を仕掛ける．システム管理者の管理外.
基本エージェント	システム管理者よる生成後，モニタマシンに常駐し IDS と連動して処理を行うエージェント.
移動エージェント	基本エージェントにより生成され，IDS の情報を元に被害先へ移動し処理を行うモバイルエージェント.

5.2　基本エージェントと IDS との連動

基本エージェントと IDS である snort-2.0.0 との間の通信にはソケット通信を用いた. そのため snort-2.0.0 のソースコードに関数 void port_cone (Packet * p) を追加した. Packet 型であるポインタ p は snort 内で定義される構造体で, snort の機能により検知

されたパケット情報を持つ．関数 port_cone( )の役割は，基本エージェントとの間にポート接続によるソケットを確立し，Packet 型へのポインタ p を引数にとり，攻撃元 IP である p->iph->ip_src，被害先 IP である p->iph->ip_dst を文字列として基本エージェントに送信する．

5.3 パケットフィルタ

パケットフィルタとは IP パケットのヘッダに含まれている情報をもとに，通信を制御する機能である．

本システムで用いているパケットフィルタ機能は Windows2000，Windows XP のサービスの一つである Routing and Remote Access を使用し，送信元 IP アドレス，発信元 IP アドレスを判断し通過／遮断する．

5.4 処理動作

本システムの各構成要素の連携した動作について述べる．システム全体の流れは図 5.1，図 5.2 に示す．

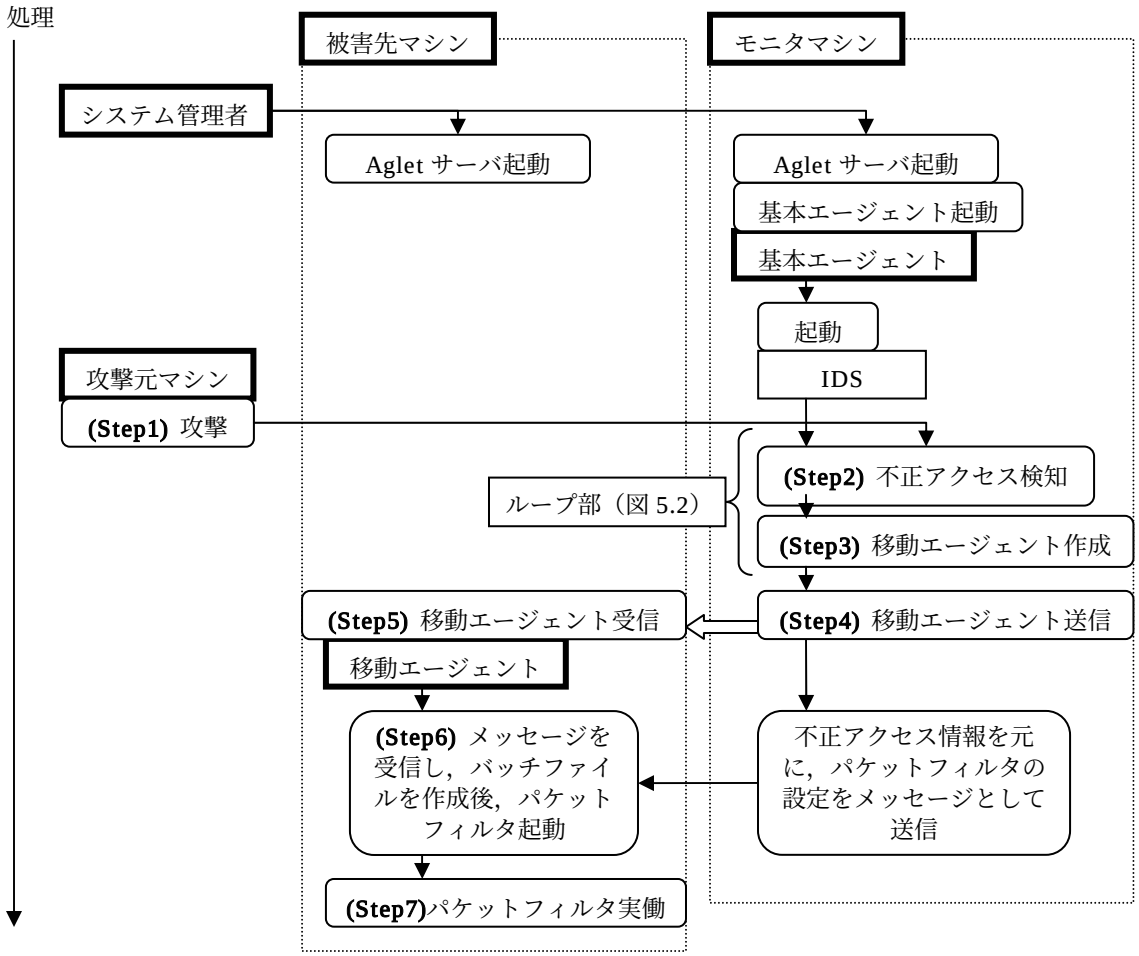

図 5.1　システム全体の処理と実装実験の計測ステップ

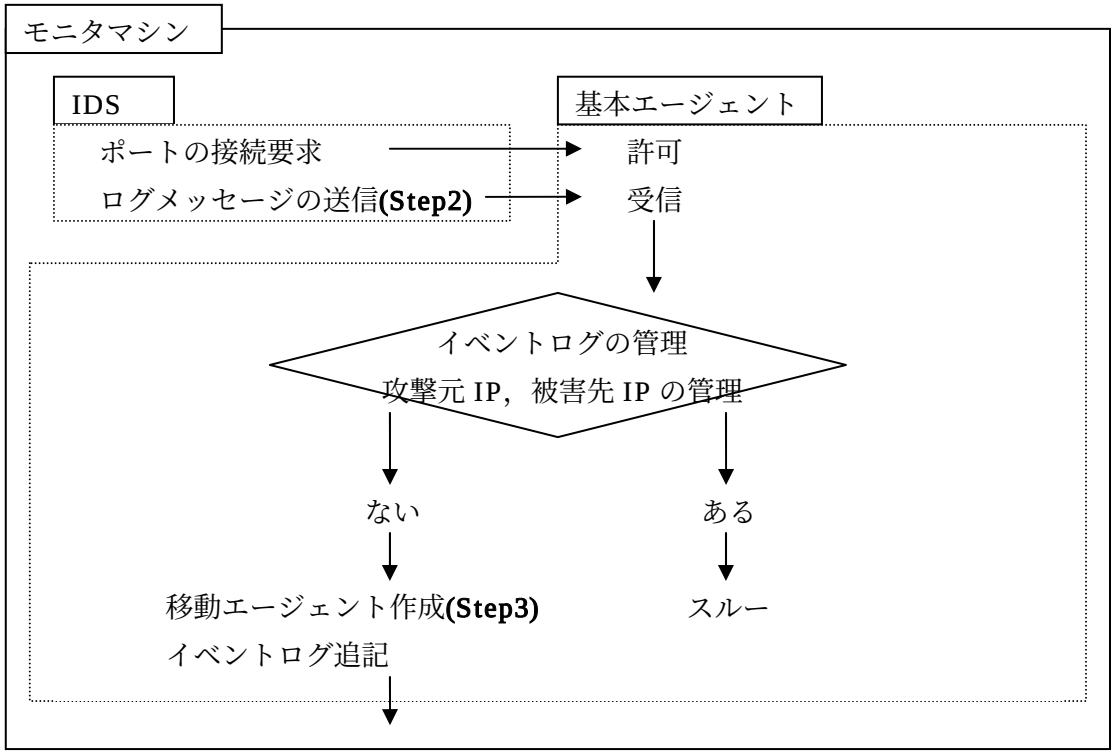

図 5.2 ループ処理

5.4.1 システム管理者の作業

システム管理者が行う処理は，本システムの起動処理を手動で行う．モニタマシン，被害先マシンに Aglet サーバを起動させ, モニタマシン上に基本エージェントを作成する. 手動で行うのはこの部分のみとなり, その後の処理は作成したエージェントが行う.

5.4.2 モニタマシンの動作

システム管理者により Aglet サーバが起動された後，基本エージェントが作成される．作成された基本エージェントにより，IDS が起動され，以降，IDS が不正アクセスを検知するまで基本エージェントは次の処理は行わない.

IDS が不正アクセスを検知した場合, 不正アクセス情報を同モニタマシン上の基本エージェントにポートを通して伝える．その後，基本エージェントにより移動エージェントが作成され，IDS が検知した情報を元に，基本エージェントにより被害先マシンの IP を認識し，移動エージェントを被害先マシンへ送信する．その後，被害先マシン上の移動エージェントにパケットフィルタコマンドを作成のため, IDS が検知した不正アクセス情報をメッセージとして送信する.

5.4.3　被害先マシンの動作

被害先マシン上では，エージェントが活動するため，システム管理者により Aglet サーバが起動されているもとする．

不正アクセスを受信したとき，リピーターハブで繋がっているモニタマシンの IDS も同時に不正アクセスを検知する．モニタマシン上の内部処理を経て送信された移動エージェントを受信する．移動エージェント受信後，基本エージェントより不正アクセスに関するメッセージが送信されるので，移動エージェントはそのメッセージをバッチファイルに記録しパケットフィルタコマンドを作成する． その後，移動エージェントはパケットフィルタを起動する．

5.4.4　攻撃元マシンの動作

被害先に攻撃用パケットを送信する．

5.4.5　基本エージェントの処理

基本エージェントはシステム管理者により，手動でモニタマシン上に作成される．作成後，基本エージェントは即座にモニタマシン上に IDS を起動させる．以降，モニタマシン上に常駐し，IDS が不正アクセスを検知するまで，次の処理を行わない．

モニタマシン上で起動する IDS が不正アクセスを検知した場合，不正アクセス情報が同モニタマシン上のポートを通して基本エージェントに伝わる．基本エージェントはモニタマシン上で移動エージェントを作成する．モニタマシン上にて，IDS により検知された不正アクセス情報を元に，被害先マシンの IP を認識し，被害先マシンへ移動エージェントを送信する．モニタマシン上にて，IDS で検地した情報を被害先マシン上の移動エージェントにメッセージとして送信する．このメッセージは被害先マシンのパケットフィルタコマンドとなる．

5.4.6　移動エージェントの処理

移動エージェントはモニタマシン上の基本エージェントにより作成される．作成される条件は IDS により不正アクセスが検知されたときである．

基本エージェントにより作成されたのち，モニタマシンから被害先マシンへ送信される．モニタマシン上の基本エージェントから，被害先マシン上にてメッセージを受信し，パケットフィルタコマンドとしてバッチファイルを作成する．作成されるバッチファイルは表 5.2 のようなもので，内容としては本システムでパケットフィルタとして使用するサービス「Routing and Remote Access」の起動，パケットフィルタの新しいルールを保存，新しいルールを適用してパケットフィルタを開始という 3 つの命令を持つ．そのバッチファイルを用いて被害先マシン上にてパケットフィルタコマンドを起動する．コマンドの起動には Java.Runtime.exec( )を使用する．

表 5.2　バッチファイル例

NET START "Routing and Remote Access" netsh routing ip delete filter "ローカル エリア接続" input 攻撃元 IP 255.255.255.255 被害先 IP 255.255.255.255 proto=ICMP type=255 code=255 netsh routing ip set filter name="ローカル エリア接続" filtertype=input action=forward

5.5 ループ処理

5.4 節の処理動作は一回の不正アクセスに対しての処理の流れになる．ここでは複数台の不正アクセスに対応するための追加した図 5.2 のループ処理について述べる．

ここで行う処理はイベントログの管理である．イベントログとは IDS より送信されてくる攻撃元 IP，被害先 IP を記録していくテキストファイルである．

基本エージェントは IDS により送信されてくる攻撃元 IP と被害先 IP の文字列とイベントログに表記されている文字列とを String クラスの equals( )メソッドを用いて比較を行う．

IDS により送信されてくる攻撃元 IP と被害先 IP がすでにイベントログに表記されている「攻撃元 IP>被害先 IP」であれば，すでに移動エージェントが作成され被害先にパケットフィルタが展開されているとして特に何も行わない. まだ表記されていないものであれば, ネットワークに対する新しい攻撃として移動エージェントの作成を行う．

第 6 章　実装実験

本章では被害先マシン 1 台について, エージェントシステムの定性的動作の有効性及びリアルタイム不正アクセス検知からの処理時間の定量的動作の両者を調べるものである実装実験と実験結果について述べ，その考察を行う.

6.1　実験環境

本システムの性能評価実験を行った環境は表 6.1，ネットワーク構成図は図 6.1 に示すとおりである．被害先マシンに流れるパケットを監視するために，モニタマシンとの接続にはリピーターハブを用いた.

表 6.1　実験環境

モニタマシン	
OS	Windows XP Pro SP1
CPU	Pentium Ⅳ 1.51GHz
RAM	256MB
被害先マシン	
OS	Windows 2000 SP4
CPU	Pentium II 300MHz
RAM	96MB
攻撃元マシン	
OS	Windows 2000 SP4
CPU	Pentium II 333MHz
RAM	512MB
LAN	100BASE リピーターハブ （CentreCOM FH505E）

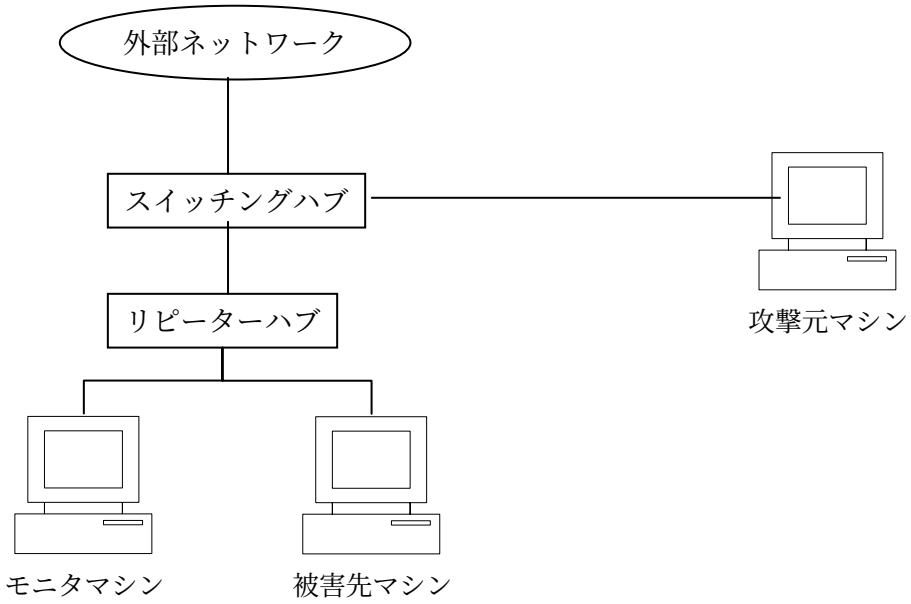

図 6.1　ネットワーク構成図

6.2　実験方法

実装実験は以下の Step を順に 11 回実施し，各 Step 間の処理時間を計測することで行われる．

各マシンの時間の同期は Simple Network Time Protocol（SNTP）を用いた（IT 用語辞典 e-Words：SNTP とは：http://e-words.jp/w/SNTP.html ）．

Network Time Protocol（NTP）とはネットワークで結ばれたコンピュータ同士で時刻を同期させるためのプロトコルで, SNTP はクライアントの時刻合わせ用途に向けて NTP を軽量化したプロトコルである．実験を行うたびに各クライアントの SNTP 設定を手動で行い時刻の同期をはかった．「net time」コマンドのオプションで SNTP サーバアドレスを指定してやると, SNTP サーバを参照しマシンの時刻が同期するようになる．今回 SNTP サーバ（時刻合わせ元）は IDS が設置されているモニタマシン（192.168.1.30）である．コマンドは表 6.2 になる．

表 6.2　攻撃元マシン，被害先マシンで時刻の同期とるコマンド

＞net time ¥¥192.168.1.30 /set /yes

処理の流れについては図 5.1 を参照されたい．ただし，ここでは単体での試験が目的であり 5.5 節（図 5.2）のループ処理を行わないものとした．

（Step１）攻撃元マシンより攻撃パケットの送信開始．今回の実験では攻撃パケットの内容は Windows2000 発信の ping とし，IDS はそのパケットを不正アクセスとして検知するルール設定がなされている．

（Step２）モニタマシンにて稼動している IDS により攻撃を検知．

（Step３）モニタマシンにて移動エージェント作成．IDS の機能により不正アクセスを検知したら，基本エージェントは移動エージェントを呼び出す．

（Step４）モニタマシンにて移動エージェント送信．移動エージェントの送り先は IDS が検知した情報を元に決められ，移動エージェントが被害先マシンに移動する．

（Step５）被害先マシンにて移動エージェント受信．

（Step６）被害先マシンにてパケットフィルタコマンドの起動. 基本エージェントは IDS の情報をパケットフィルタのルール設定にするため, 移動エージェントにメッセージとして渡し，移動エージェントがパケットフィルタを起動する．

（Step７）被害先マシンにてパケットフィルタ動作開始．パケットフィルタは攻撃元マシン IP から被害先マシン IP へのパケットを弾く．

6.3　実験結果

Step１～Step７までを１１回実施した結果の平均処理時間を表 6.3，図 6.2 に示す．

表 6.2　計測結果

	1 回目	2～11 回目平均
①Step1-2	0.37×10^{2}	0.51×10^{2}
②Step2-3	0.51×10^{3}	0.29×10^{3}
③Step3-4	0.47×10^{2}	0.39×10^{2}
④Step4-5	0.16×10^{4}	0.14×10^{4}
⑤Step5-6	0.26×10^{3}	0.19×10^{3}
⑥Step6-7	1.22×10^{4}	0.47×10^{4}
全体 Step1-7	1.47×10^{4}	0.67×10^{4}

単位はミリ秒．

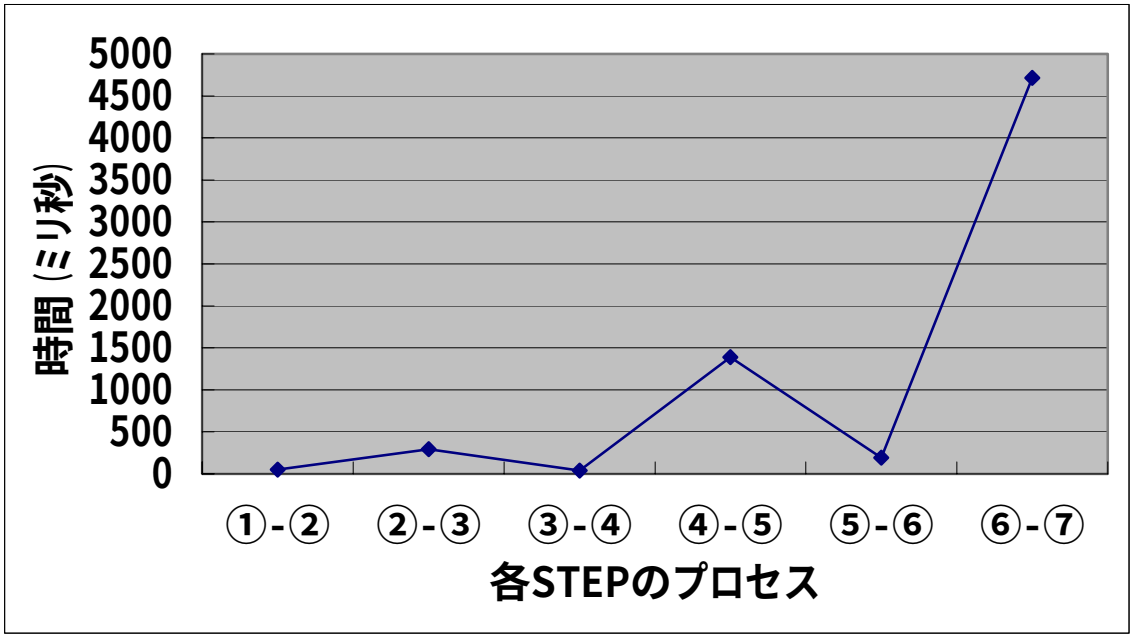

図 6.2　２回目～１１回目の各 Step 間の平均処理時間

6.4　エージェント非稼動時の各マシンステータスの調査

本システムでエージェントが起動するときのモニタマシンの CPU 使用率，被害先の CPU 稼働率，被害先マシンの受信トラフィックの調査を行った．

6.4.1　調査方法

調査に用いたツールは CRNMonitor0.99 である．これは CPU，メモリ，ネットワークをそれぞれ監視しマルチプロセッサ対応，各種ログ出力機能を持つ．ログの記録は 1 秒ごとに行い，それぞれを図 6.3，図 6.4，図 6.5 に示す．なお移動エージェントのサイズは 4Kbyte 程度である．

CRNMonitor0.99：http://www.vector.co.jp/soft/dl/win95/hardware/se331669.html

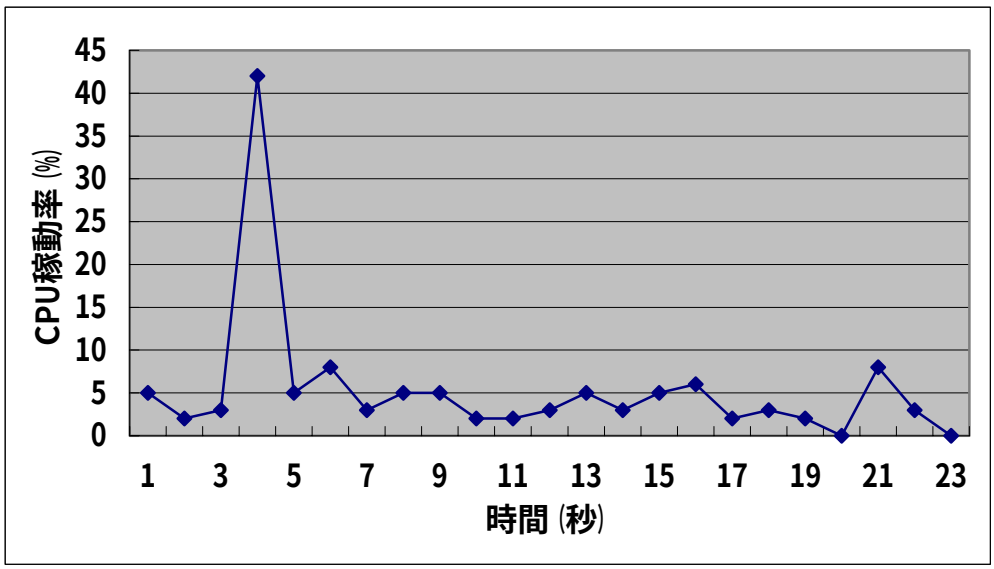

図 6.3　モニタマシンの CPU 使用率

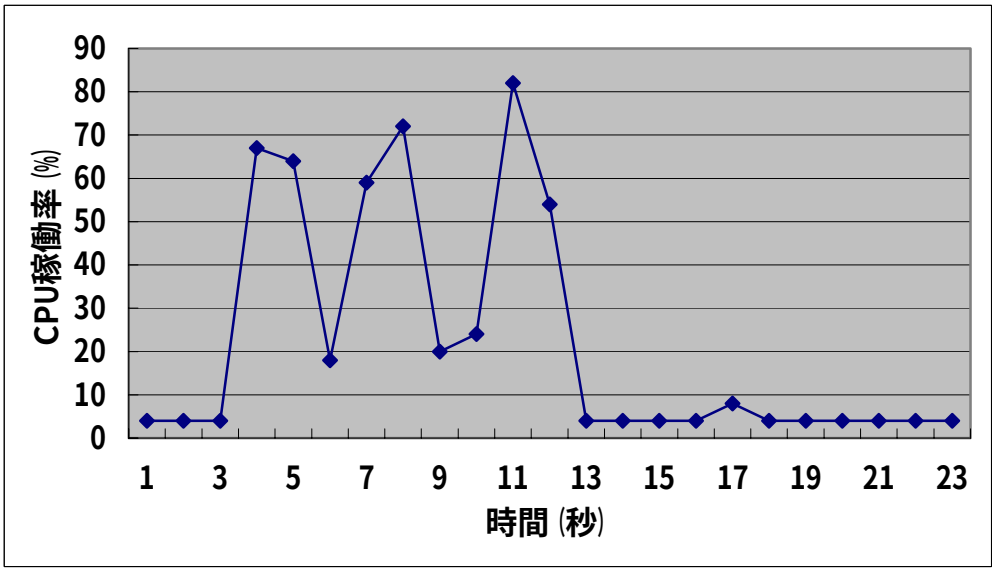

図 6.4　被害先の CPU 稼働率

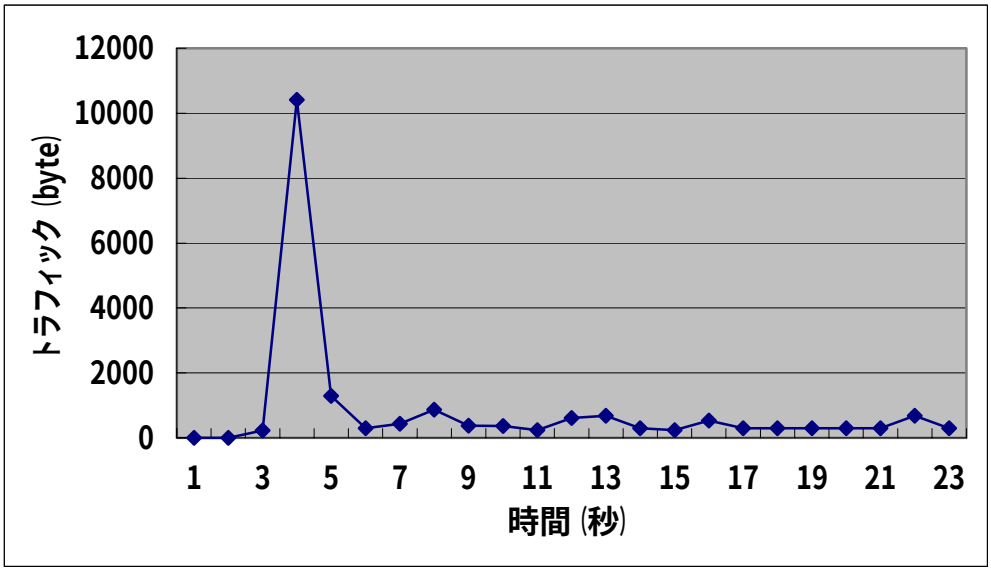

図 6.5　被害先マシンの受信トラフィック

6.4.2　調査結果

不正アクセスのパケット検知からの移動エージェント作成開始までは 1 秒とかからないが (6.3 節), マシンステータスの変化を見るため不正パケット受信前よりログを取っているためで 3 秒あたりが移動エージェント作成となっている.

そのため, 基本エージェントが移動エージェント作成を行い, 被害先マシンに送信を行う 3〜5 秒あたりで, モニタマシンの CPU 使用率, 被害先の CPU 稼働率, 被害先マシンの受信トラフィックのすべてにおいて上昇が見られる.

6.5　エージェント非稼動時の被害先マシンステータスの調査

6.4 節の比較対照としてエージェント非稼動時の被害先マシンステータスの調査を行った.

6.5.1　調査方法

被害先マシンに対し 0.1 秒毎, 0.2 秒毎, 0.5 秒毎, 1 秒毎に ping パケット 1 つを 1 回送信し, それを 20 秒間続けた状態と何も行っていない状態での受信トラフィックと CPU 使用率を 6.4.1 節と同様に CRNMonitor0.99 を用いて調査した. ログの記録は 1 秒毎に行い, その結果を図 6.6, 図 6.7 に示す.

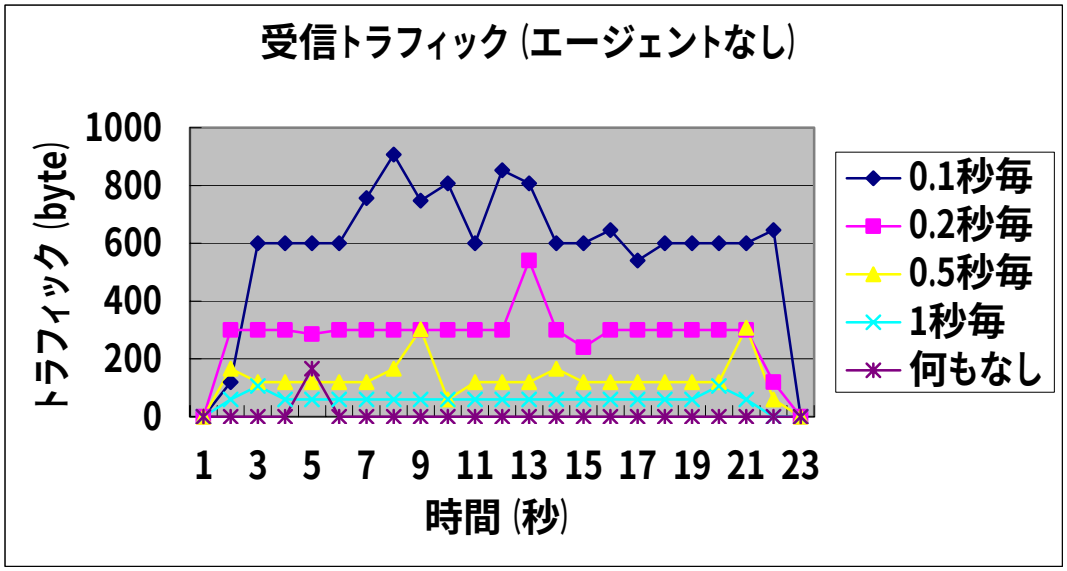

図 6.6　エージェント非稼動時の被害先マシン受信トラフィック

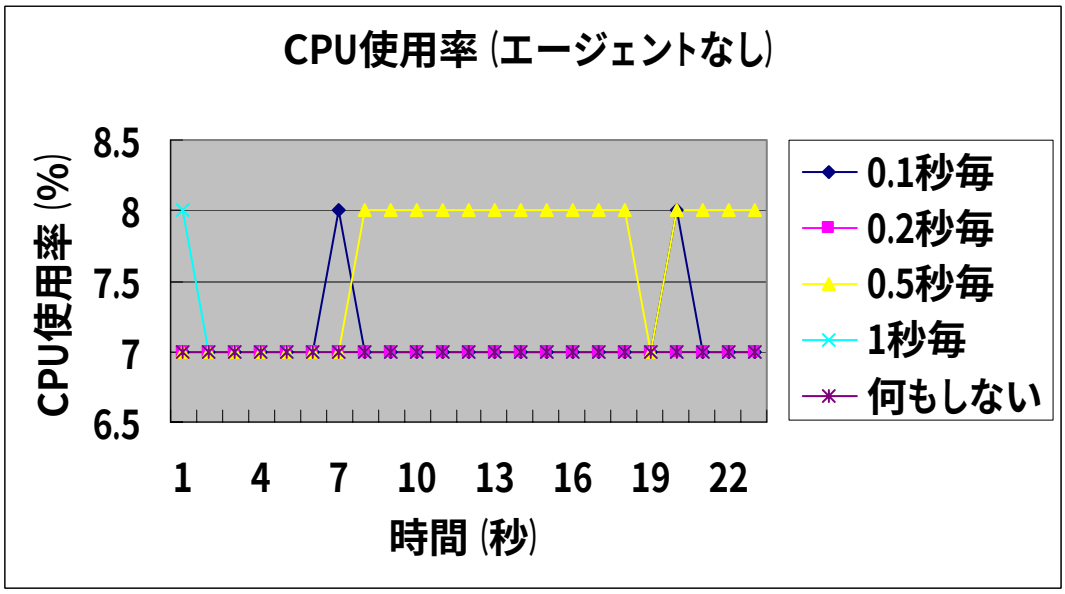

図 6.7　エージェント非稼動時の被害先マシン CPU 使用率

6.5.2　調査結果

現在の実験程度(0.2 秒毎の ping)の負荷では CPU 使用率に変化は見られなかった. 6.4 節のエージェントが被害先マシンへ移動しパケットフィルタ展開にて被害先マシンの CPU 使用率が大幅に上昇した（最大 82%）のは，パケットフィルタに使用する

Routing and Remote Access の処理が重いためである．参考までに被害先マシンよりスペックの良いモニタマシン上でパケットフィルタを起動した際の CPU 使用率を調査したが（図 6.8），こちらでもかなりの上昇が見られた．

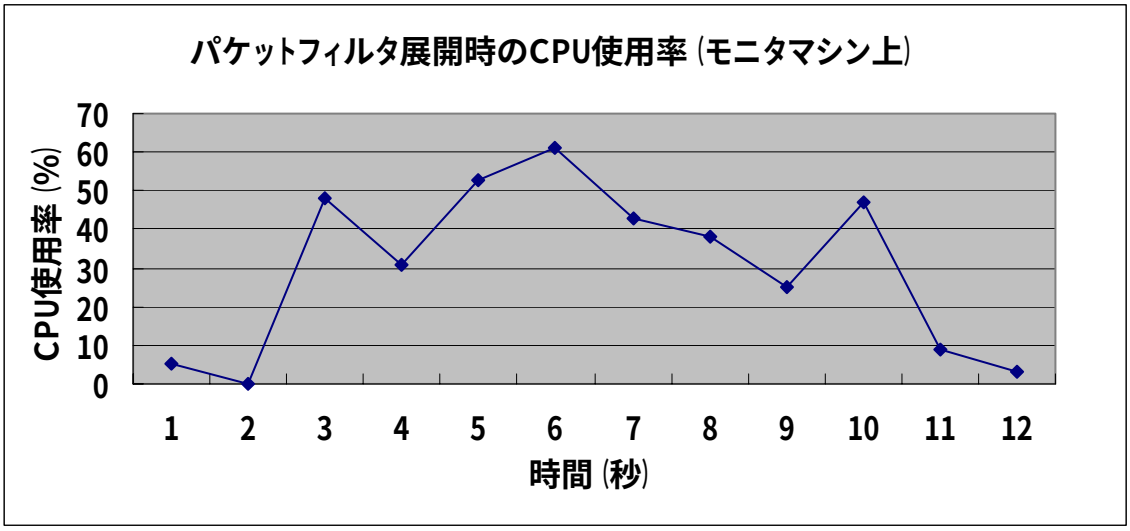

図 6.8　モニタマシン上でのパケットフィルタ展開時の CPU 使用率

6.6　考察

6.3 節の時間計測では１回目と２回目以降では大きく時間に差がある．これは２回目以降，ホットスタートにて実験を行っているためで，キャッシュにより高速に呼び出されているためである．

Step４－５間の処理時間については，移動エージェントの移動時に攻撃元マシンより攻撃用パケットが送信されているため，被害先マシンの処理能力が低下しているか，または，リピーターハブを用いているため，ネットワーク内のトラフィックが大きくなり転送速度が低下しているためである．

Step６－７間の処理時間は，Windows のサービスである Routing and Remote Access を使用したため，高速起動に対応していないことが原因であろう．専用のフィルタリング機能作成や被害先マシンの性能向上といった対策が可能であろう．

第 7 章　複数台への対応実験

本章では第 6 章でのシステムに 5.5 節（図 5.2）の内容を追加し，エージェントシステムの複数台への対応性を調べた実装実験と実験結果について述べ，その考察を行う．

7.1　実験環境

複数台への対応を調べるために行った実験環境は表 7.1，ネットワーク構成は図 7.1 に示すとおりである．被害先マシン①②に流れるパケットを監視するために，モニタマシンとの接続にはリピーターハブを用いた．なお，モニタマシンは被害先マシン②を兼用している．

表 7.1　実験環境

モニタマシン	
OS	Windows XP Pro SP1
CPU	Pentium Ⅳ 1.51GHz
RAM	256MB
被害先マシン①	
IP	192.168.1.60
OS	Windows 2000 SP4
CPU	Pentium II 300MHz
RAM	96MB
被害先マシン②（モニタマシン兼用）	
IP	192.168.1.30
OS	Windows XP Pro SP1
CPU	Pentium Ⅳ 1.51GHz
RAM	256MB
攻撃元マシン①	
IP	192.168.1.40
OS	Windows XP SP2
CPU	Celeron 2.8GHz
RAM	512MB
攻撃元マシン②	
IP	192.168.1.102
OS	Windows 2000 SP4

CPU	Pentium II 333MHz
RAM	512MB
攻撃元マシン③	
IP	192.168.1.201
OS	Windows 2000 SP4
CPU	Celeron 1.7GHz
RAM	256MB
LAN	100BASE リピーターハブ （CentreCOM FH505E）

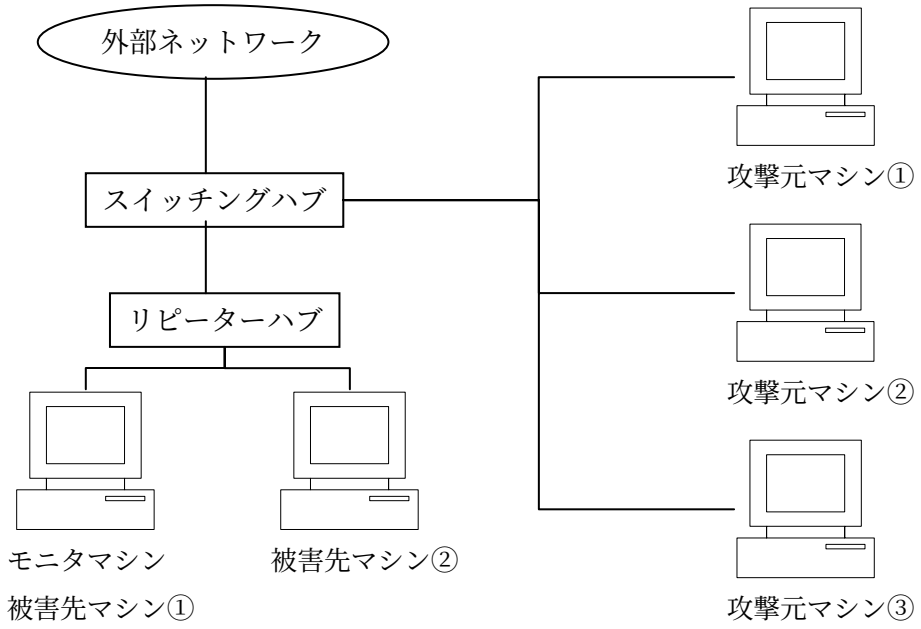

図 7.1　ネットワーク構成

7.2　実験方法

モニタマシン上に基本エージェント作成後，攻撃元マシンから被害先マシンに対して Windows のデフォルト ping（1 秒間隔で 4 回パケット送信）を表 7.2 の順に送信.

表 7.2　パケット送信順

順番	送信元	送信先	補足

1	攻撃元マシン① 192.168.1.40	被害先マシン① 192.168.1.60	
2	攻撃元マシン② 192.168.1.102	被害先マシン① 192.168.1.60	
3	攻撃元マシン③ 192.168.1.201	被害先マシン① 192.168.1.60	
4	攻撃元マシン① 192.168.1.40	被害先マシン② 192.168.1.30	
5	攻撃元マシン② 192.168.1.102	被害先マシン② 192.168.1.30	
6	攻撃元マシン③ 192.168.1.201	被害先マシン② 192.168.1.30	
7	攻撃元マシン① 192.168.1.40	被害先マシン① 192.168.1.60	1 と同じ
8	攻撃元マシン② 192.168.1.102	被害先マシン① 192.168.1.60	2 と同じ
9	攻撃元マシン③ 192.168.1.201	被害先マシン① 192.168.1.60	3 と同じ
10	攻撃元マシン① 192.168.1.40	被害先マシン② 192.168.1.30	4 と同じ
11	攻撃元マシン② 192.168.1.102	被害先マシン② 192.168.1.30	5 と同じ
12	攻撃元マシン③ 192.168.1.201	被害先マシン② 192.168.1.30	6 と同じ

7.3 実験結果

移動エージェントが作成されたのは順番 1，2，3，4，5，6 時に各 1 個．その作成された移動エージェントが送信された先は順番 1，2，3 の時は 192.168.1.60，順番 4，5，6 時は 192.168.1.30 であった．

また実験後の基本エージェントにより作成されたイベントログは表 7.3 となった．

表 7.3　動作確認実験後のイベントログ

192.168.1.40>192.168.1.60
192.168.1.102>192.168.1.60
192.168.1.201>192.168.1.60
192.168.1.40>192.168.1.30
192.168.1.102>192.168.1.30
192.168.1.201>192.168.1.30

7.4　考察

表 7.3 で攻撃元 IP ＞ 被害先 IP が同じパケットにはイベントログが残っていないがことより，重複した移動エージェントを作成せずに，エージェントの元で活動している IDS の監視下にある複数のマシンに被害がおよんだ場合でも対応することができることが判った.

しかし，実践レベルでは速度向上，デバグ処理，パケットフィルタルールの細分化などの問題がある.

第 8 章　終論

8.1　問題点

ここでは現状での問題点について述べる.

8.1.1　システムについて

現状では移動エージェントはパケットフィルタ起動後に行動を起こさないが, パケットフィルタ後に攻撃の停止を認識できればパケットフィルタを解除するといった機能が必要である. 反面, ネットワーク管理者に権限が集中するため悪意を持った管理者がエージェントシステムを悪用し，結果，DOS 攻撃などの踏み台にされてしまう可能性がある.

エージェントの移動のためポートを 1 つ開けておく必要があり, このポートが攻撃対象となる可能性ある. 各マシン上で Aglet サーバが起動していなければ, システムが起動しない. 外部からの攻撃で Aglet サーバを落とすためには，Aglet の活動ポート番号を知り，Java の特定のプロセスを落とすといった手順があげられる.

移動エージェントによりフィルタプログラムをバッチファイルに保存し, 実行する行動を行うため, Java のセキュリティポリシーをあらかじめ変更しておく必要があり（表 8.1），この変更点も攻撃対象となる.

8.1.2　複数台への対応について

複数台への対応化起動前にイベントログファイルに対して write, read ができるよう java のセキュリティポリシーを変更する必要がある.（表 8.1）

2 台の攻撃元マシンから同じ被害先に同時に不正アクセスがあった場合, 移動エージェントを 2 つ作成する. そのとき 1 つ目の移動エージェントが被害先へ移動しパケットフィルタを展開していると，2 つ目の移動エージェントの移動できない.

基本エージェントが IDS からのパケットメッセージを取りこぼす可能性があり，原因としては Snort－基本エージェント間の並列処理がうまくいっていない, または基本エージェントの処理が間に合っていないためであると考えられる.

表 8.1　本システムで変更する java のセキュリティポリシー

モニタマシン
// IDS－基本エージェント間の通信 permission java.net.SocketPermission "モニタマシン：ポート番号", "accept,listen,connect,resolve";

//被害先マシンとの通信 permission java.net.SocketPermission "モニタマシン:ポート番号", "accept,listen,connect,resolve";
//IDS の起動 permission java.io.FilePermission "snort.exe へのパス", "execute";
//イベントログの入出力 permission java.io.FilePermission "イベントログへのパス", "write,read";
被害先マシン
//モニタマシンへの通信 permission java.net.SocketPermission "モニタマシン:ポート番号", "accept,listen";
//パケットフィルタ実行 permission java.io.FilePermission "パケットフィルタ用バッチファイルへのパス","write,execute";

8.2 まとめ

本研究では，モニタマシン上の IDS が不正アクセスを検知した場合，システム内のエージェントが行動を起こし，被害先マシン上にパケットフィルタを展開し，不正アクセスを防ぐことができた．これにより，ネットワーク内のモニタ可能なマシンすべてを対象にすることができると考えられる．また内部からの攻撃に対しても有効であるといえる．

8.3 今後の展望

現段階のシステムでは複数台へのマシンへの対応化ができると言っても，実践レベルで使用するためには IDS との並列処理での速度向上，デバグ作業を行う必要がある．またエージェントの機能が限られており，パケットフィルタルールの細分化を行い，特定 IP アドレスパケットフィルタのみでなく，ポート別，プロトコル別のフィルタなどの機能の追加が今後の課題である．

謝辞

本研究にあたり，最後まで熱心な御指導をいただきました田中章司郎教授には，心より御礼申し上げます．また，田中研究室の皆さんには，本研究に関して数々の御協力と御助言をいただきました．厚く御礼申し上げます．

なお，本論文，本研究で作成したプログラム及びデータ，並びに関連する発表資料等のすべての知的財産権を，本研究の指導教官である田中章司郎教授に譲渡致します．

参考文献・資料

[1] Gangadharan, M. Kai Hwang: Intranet security with micro-firewalls and mobile agents for proactive intrusion response, *Proceedings 2001 International Conference on Computer Networks and Mobile Computing,* pp.325-332(2001).

[2] Kai Hwang, Gangadharan, M.: Micro-firewalls for dynamic network security with distributed intrusion detection, *Proceedings IEEE International Symposium on Network Computing and Applications,* pp.68-79(2001).

[3] 渡辺勝弘，伊原秀明：不正アクセス調査ガイド，オーム社，2002

[4] Brian Caswell，James C. Foster，Ryan Russell，Jay Beale，Jeffrey Posluns（2003）：Snort 2.0 Intrusion Detection．渡辺勝弘，鹿田幸治，Snort 翻訳プロジェクト 訳『Snort 2.0 侵入検知―オープンソースソフトウェアで構築する実践的な侵入検知システム』ソフトバンク，2004

[5] 新谷虎松：java による知能プログラミング入門，コロナ社，2002