平成 14 年度　卒業論文

ネットワーク侵入検査機能の改良について

２００３年2月１２日

島根大学 総合理工学部 数理・情報システム学科

計算機科学講座（田中研究室）

ｓ９９４１３－G　　長田　昌訓

目次

1章 はじめに

近年、一般家庭にも常時接続が普及し、不正アクセスを受けるケースが増えてきている。この卒業研究では、インターネットからの攻撃や侵入などの不正アクセスから自分の PC または LAN を守るためにリアルタイムな侵入検査システムの改良を目的とする。

（渡辺勝弘、2002）によれば、ネットワークモニタリングの目的としては以下の３つが考えられる。

１、侵入者の特定・被害状況の記録
ログに残された記録から侵入者を特定したり、被害状況の確認を行う。

２、被害の防止および軽減
システムへの侵入を受けた場合には早期に侵入を発見し、すばやく対処することで、被害の発生の防止、軽減を図る。

３、システムのセキュリティに対する脅威の評価
システムが受けている攻撃について監視と記録を実施し、どの程度の脅威が存在しているかを確認する。ここで得られたデータをもとに、以後のセキュリティ計画の立案や設定などを行う。

本研究では侵入検知システム（Intrusion Detection System ：IDS）である snort を用いて、侵入を受けた際にログに記録するだけではなく、簡単なメールとしていち早く管理者に知らせるシステムを作成した。

2章　侵入検査システムの導入

2.1　侵入検査システム（Intrusion Detection System）とは

コンピュータやネットワークに対する不正なセキュリティ侵害や攻撃の検出を行うシステムである。中でも、各ホストコンピュータで稼動し、そのホストコンピュータで不正な行為がなされていないかを監視するシステムを「ホストベース IDS」、ネットワーク上に流れるパケットを監視・調査するシステムを「ネットワークベース IDS」と区別される。ネットワークを経由してアクセスしたのであれば、必ずネットワーク上に通信が発生する。その通信をうまく傍受することができれば、誰がどこから自分のコンピュータのどのポートと通信しているのかを知ることができる。

2.2　侵入検査システム稼動に必要なツール

侵入検査システムである snort は UNIX や Linux で作られたため、Windows で snort を動かすために必要なツールを紹介する。

2.2.1　snort

Snort は Marty Roesch を中心にして開発され、GNU General Public License のもとでフリーウェアとして公開されているネットワークベース IDS で、その機能は入力パケットを解析し予め登録されているルールとパターンマッチングを行いシステムへの侵入を検知するというもので、Linux を始めとして多くのプラットホームで稼動する。

Snort のソースは以下の URL から入手できる。

http://www.snort.org/

2.2.2　パケットキャプチャドライバ

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。

http://www.tcpdump.org/release/

また Win32 版の Snort を実行するには「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。

http://netgroup-serv.polito.it/winpcap/

2.2.3　snort.panel

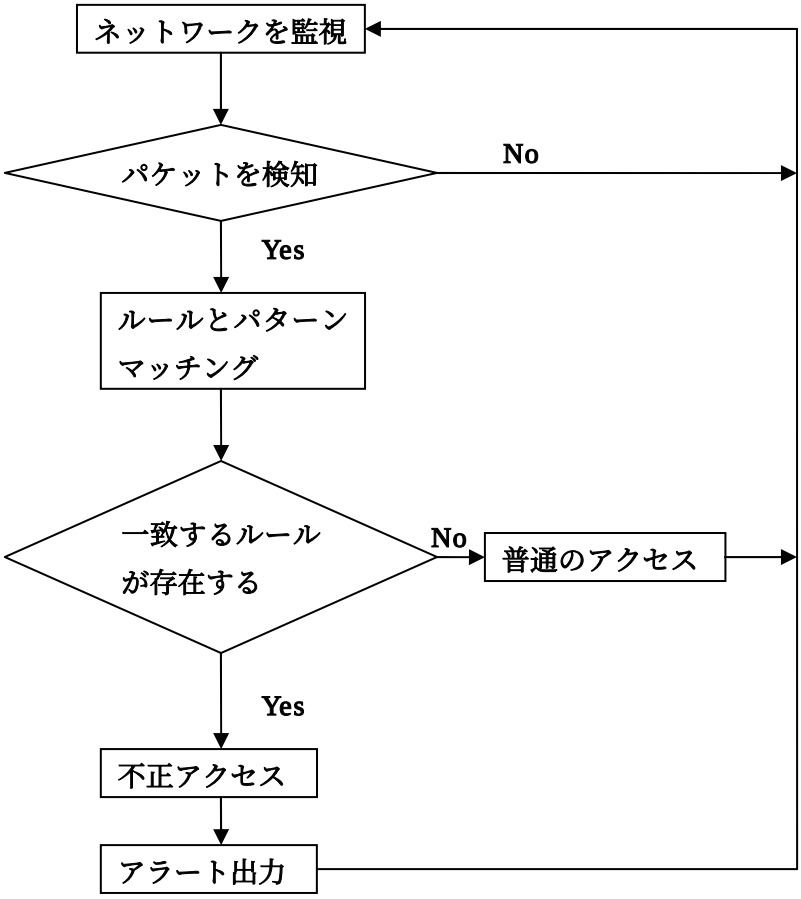
(米)Xato Network Security が開発した snort を Win32 版のＧＵＩ（graphical user interface）で利用するためのツール。以下より入手。

http://www.xato.net/files.htm

3 章　Windows 版 Snort 1.7 の機能

3.1　snort のフローチャート

snort のフローチャートは、通信パターンをルールと比較して一致するパターンを発見すると、それを警告としてログに記録するというものである。



3.2　ルールファイル

引用文献：[1]

Snort ではルールファイルを読み込んで動作するようになっている。

以下はWindows用snort1.7に付属するルールファイルを検出するジャンル毎にまとめたものである。

snort.conf　　ルールセットのメインファイル
- ｜－backdoor-lib　　バックドアとの通信の検出。
- ｜－ddos-lib　　DDoS 関連の通信および攻撃の検出。
- ｜－finger-lib　　さまざまなクライアントからの finger の検出。
- ｜－ftp-lib　　ftp に対する通信や攻撃の検出。
- ｜－misc-lib　　リモート管理ツールやその他の検出。
- ｜－netbios-lib　　netbios による不正利用の検出。
- ｜－overflow-lib　　バッファ・オーバーフロー攻撃の検出。
- ｜－ping-lib　　さまざまなクライアントからの ping の検出。
- ｜－rpc-lib　　rpc に対する通信や攻撃の検出。
- ｜－scan-lib　　ステルス系ポートスキャン等の検出。
- ｜－smtp-lib　　smtp に対する通信や攻撃の検出。
- ｜－telnet-lib　　telnet に対する通信や攻撃の検出。
- ｜－web-lib　　www に対する通信や攻撃の検出。
- ｜－webcf-lib　　ColdFusion に対する通信や攻撃の検出。
- ｜－webcgi-lib　　cgi に対する通信や攻撃の検出。
- ｜－webfp-lib　　FrontPage に対する通信や攻撃の検出。
- ｜－webiis-lib　　IIS に対する通信や攻撃の検出。
- ｜－webmisc-lib　　その他の www 関連サービスへの通信や攻撃の検出

snort.conf は他の*-lib ファイルをインクルードする形式で記述されている。従って１つのジャンルをまるごと無効にする場合は、該当の include 行をコメントアウトするだけでよい。ping 関連の検出を無効にするのであれば、snort.conf ファイルで include されている部分をコメントアウトする。

include ping-lib　→　#include ping-lib

3.2　ルール書式

例：ping-lib ファイルの１行

```
alert icmp any any -> $HOME_NET any (msg:"IDS159 - PING Microsoft Windows";
content:"|6162636465666768696a6b6c6d6e6f70|";itype:8;depth:32;)
```

snort のルールは、（　）でくくられた部分とそれ以外の部位に分けられ、それぞれ前半部をルールヘッダ、後半部の括弧でくくられた部分をルールオプションと呼ぶ。

3.2.1　ルールヘッダ部

ルールヘッダ部は以下のように構成されている。

1. ルールに記述された条件に合うパケットを受信した場合に、そのルールが起こすべき行動。
   alert：ルールオプションに従ってアラートを出力
   log：パケットをログディレクトリ内に記録する
   pass：パケットを通過させる
2. 監視するプロトコル（ＴＣＰ、ＵＤＰ、ＩＭＣＰの３種）
3. 通信元ＩＰアドレス、ポート番号
4. データの流れ（左から右は->、右から左は<-、双方< >）
5. 通信先のＩＰアドレス、ポート番号を記述する。

なお、書式は次のとおりである。

＜行動＞　＜プロトコル＞　＜IP アドレス＞/＜ネットマスク＞　＜ポート番号＞ **＜データの流れ＞　＜IP アドレス＞/＜ネットマスク＞　＜ポート番号＞**

3.2.2 ルールオプション部

ルールオプション部では通信内容を検査するためのＩＰアドレス・ポート情報以外の詳しい条件を記述する。書式は以下のように、:（コロン）で区切られたキーワードと値を記述し、さらに；(セミコロン）で区切って、続くキーワードと値を記述する。最大で 15 個までのキーワードを記述することができる

（＜キーワード１＞：＜値１＞；＜キーワード２＞：＜値２＞；……；）

ルールオプション部に記述できるキーワードは以下のとおり。

mgs　：ルールにマッチした場合、アラートとログにメッセージを出力する。
logto　：指定したファイルにパケットログを記録する。
ttl　：TTL（Time To Live）を評価する。
tos　：TOS（Type Of Service）を評価する。
id　：ID（IDentification）フィールドを評価する。
ipo　：指定された IP Option が含まれているか評価する。
fragbits：フラグメント（Fragmentation Bits）を評価する。
dsize　：パケットペイロードのサイズを評価する。
content：content オプションで指定されたデータと、送られてくるデータとのマッチングを行い、マッチした場合にルールヘッダ部の行動で指定された動作を行う。｜（パイプ）で囲まれているのは１６進数のバイナリデータ。
content-list：content と同様の動作を、データパターンリストを使用して行う。
flags　：TCP コードビット（TCP Flags）を評価する。
seq　：TCP のシーケンス番号（TCP Sequence Number）を評価する。
ack　：TCP の応答確認番号（TCP Acknowledgement Number）を評価する。
itype　：ICMP ヘッダ内の type の値を評価する。
icode　：ICMP ヘッダ内の code の値を評価する。
icmp_id　：ICMP ECHO の ID フィールドの値を評価する。
icmp_seq：ICMP ECHO のシーケンス番号（Sequence Number）の値を評価する。
offset　：パケットペイロード内で content のマッチングを開始する位置を指定する。
depth：content オプションとともに使用し、content オプションにより指定されたデータのマッチングを行う範囲をオクテット値で指定できる。
nocase　：content で指定された文字列のセンシティビティを低くする。
session　：アプリケーションレイヤのデータを記録する形式を指定する。
rpc　：RPC サービスのトラフィックに対して、application、proceedure、version がマッチするか評価する。
resp　：ルールにマッチしたセッションを強制的に切断する。
react　：ルールにマッチしたセッションをブロックする。

４章　追加した機能

snort は本来不正アクセスを受けた際のアラートをログに記録する機能しかない。そこでアラートをいち早く管理者に知らせるメール送信機能を追加する。

4.1　snort の改良によるメッセージファイルへの書き出し

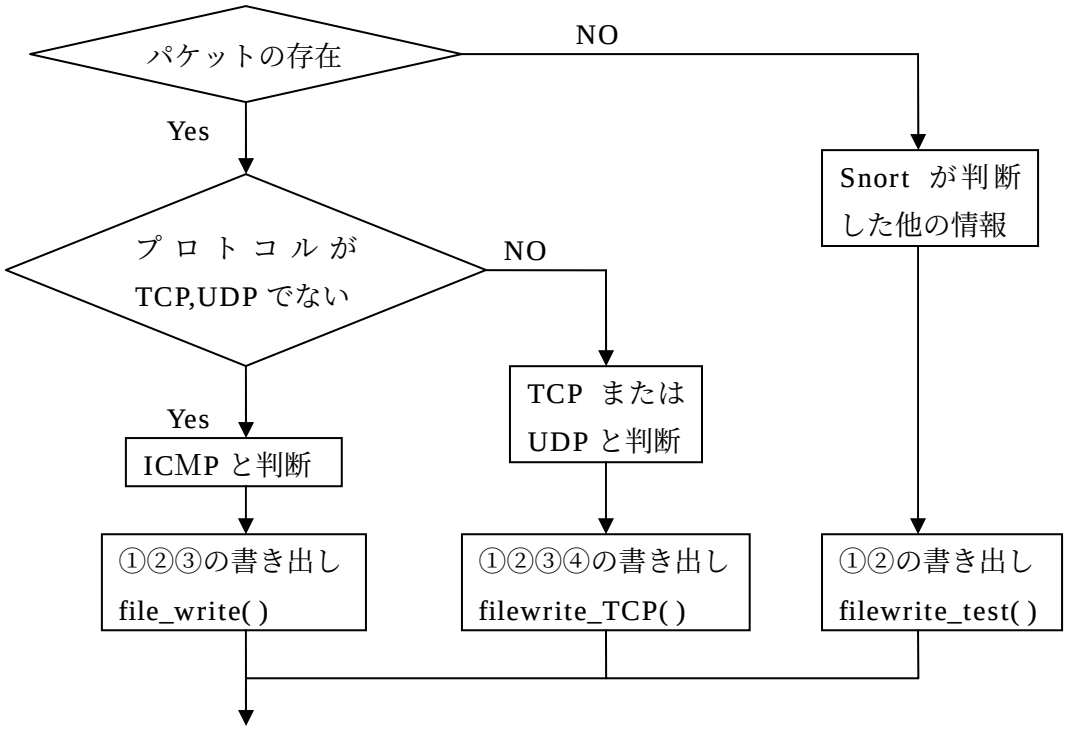
メールとして送信するときはバイトサイズを小さくするために必要な情報だけをメール本文とすることが望ましい。

必要な情報とは以下のものである。

① 不正なパケットを検知した日時
② 検知したパケットがどのようなものであるかのメッセージ
③ 送信者のＩＰ、受信者のＩＰ
④ パケットタイプがＴＣＰ，ＵＤＰであればポート番号

また、必要な情報を採取するためのプログラムを組み込む位置だが、「アラートに出力するメッセージが必ず存在する」という条件が必要となってくる。そのことを考慮すると、アラートを出力している関数内に組み込めば問題は解決する。

●メッセージファイルへの書き出しフローチャート



●概要

snort ではオプション設定によりアラート出力の詳細度を変えることができ、以下の２つの関数がアラート出力の関数となっている。

Full（詳細アラート出力と設定）時は log.c ファイルの関数 AlertFull( )
Fast（簡易アラート出力と設定）時は log.c ファイルの関数 AlertFast( )

上記のようなアラート出力処理を行っているので、それぞれの関数内にメール用ファイルに簡易メッセージを書き出す file_write( )、filewrite_TCP( )、filewrite_test( )関数を組み込む。

●関数の説明

・ file_write( )

ICMP 用のメッセージ書き出し関数。

引数には、パケット情報が格納してある Packet 構造型ポインタ p、検知した時間 timestamp、メッセージ mes の３つの値をとる。

メール用ファイルに書き出す情報。

① 時間(timestamp)
② メッセージ(mes)
③ 発信元 IP(p->iph->ip_src)　宛先 IP(p->iph->ip_dst)

（ただし、③については「192.168.1.100」という形式で出力したいので inet_ntoa(p->iph->ip_src)といった関数を用いる。）

・ filewrite_TCP( )

TCP、UDP 用のメッセージ書き出し関数。

引数は file_write( )と同じ。

メール用ファイルに書き出す情報は①②③に追加して

④ 発信元ポート(p->sp)　宛先ポート(p->dp)

・ filewrite_test( )

その他（接続回数など）から snort が検知したときのメッセージ書き出し関数。

引数は検知した時間 timestamp、メッセージ mes の２つ。

メール用ファイルに書き出す情報はパケットに関する情報が存在しないので①②となる。

4.2　メールの送信

引用文献：[2]

4.2.1　SMTP

SMTP(Simple Mail Transfer Protocol)はメールサーバ間でのメッセージの交換に利用されるプロトコルである。メールの送り元（送信側）がクライアント、受け取り先（受信側）がサーバとなる。サーバは 25 番ポートで接続を待ち、クライアントはサーバの 25 番ポートにコマンドを送る。

表 1 に SMTP の基本コマンド、表 2 に SMTP 返信コードを示す。

コマンド	パラメータ	概要
HELO	ドメイン名	通信の開始、送信側ドメインの通知
MAIL FROM :	送信ユーザ名	受信者の指定
RCPT TO :	宛先ユーザ名	受信者の指定
DATA	メッセージデータ	メール本文のデータ送信開始
RSET	（なし）	送信手順を初期化
QUIT	（なし）	SMTP 接続を終了

表１　SMTP の基本コマンド

返信コード	内容
211	システム状態の表示
220	サーバの準備完了
221	SMTP 接続終了
250	要求は正常に完了
251	メールの転送処理
354	メール本文の入力開始
421	サービスを提供できないため、接続終了
451	ローカルで問題が発生したため、処理を中断
452	記憶容量不足のために、処理を中断
500	文法の誤り、コマンド構文エラー
501	文法の誤り、パラメータや引数のエラー
503	コマンドの順序エラー
550	メールボックスが使用不能で要求は実行できない
551	ユーザがサーバマシン上には存在しない

表２　SMTP の応答コード

4.2.2　メール転送手順

図１にメールの転送手順を示す。

HELO コマンドでセッションが開始され、MAIL コマンドで発信元アドレス、次の RCPT コマンドによってあて先が指定される。サーバ側からは各コマンドに対する応答コード返される。

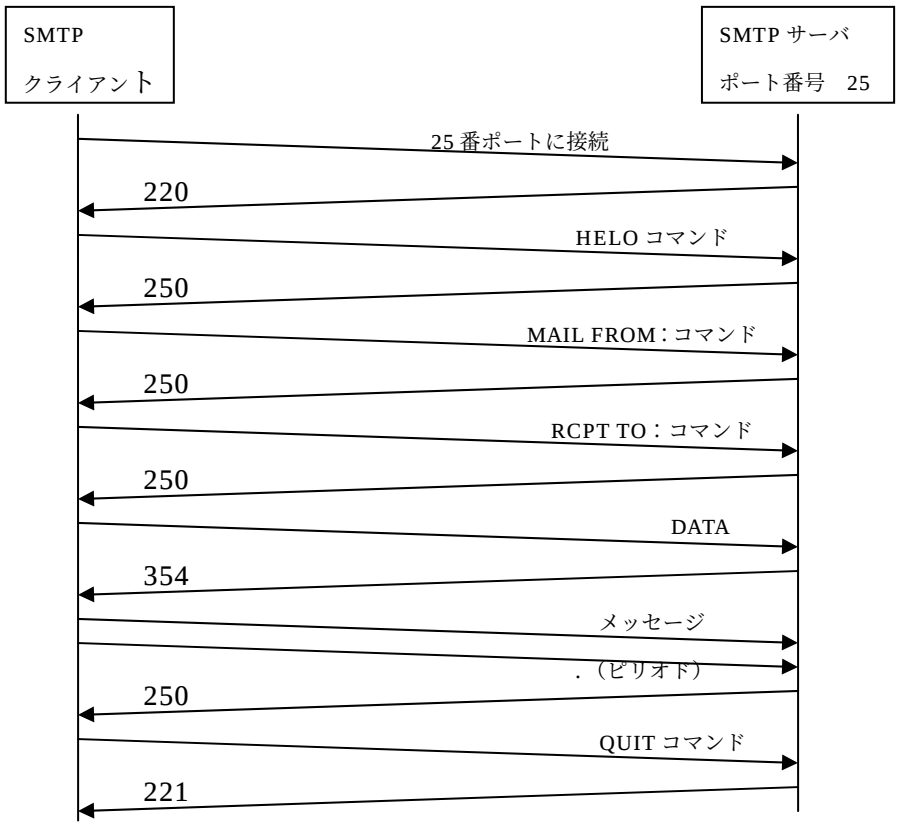


図１　メール転送手順

4.2.3 WinSock

引用文献：[3]

ソケットは UNIX から来ているので、UNIX での使用が前提となっている。Windows でネットワークプログラムを制作する場合、標準的に使用されているのが Winsock と呼ばれるソケットライブラリである。Microsoft Visual C++ を使用しているのであれば、winsock.h というヘッダファイルにすべての構造体や関数などが定義されている。

WinSock を使ったネットワークプログラムの手順と関数。
①WinSock 初期化　　WSAStartup( )
②ソケット作成　　socket( )
④サーバに接続する　　connect( )
⑤データを送受信する　　send( ) 、recv( )
⑥ソケットを閉じる　　closesocket( )
⑦WinSock を解放する　　WSACleanup( )

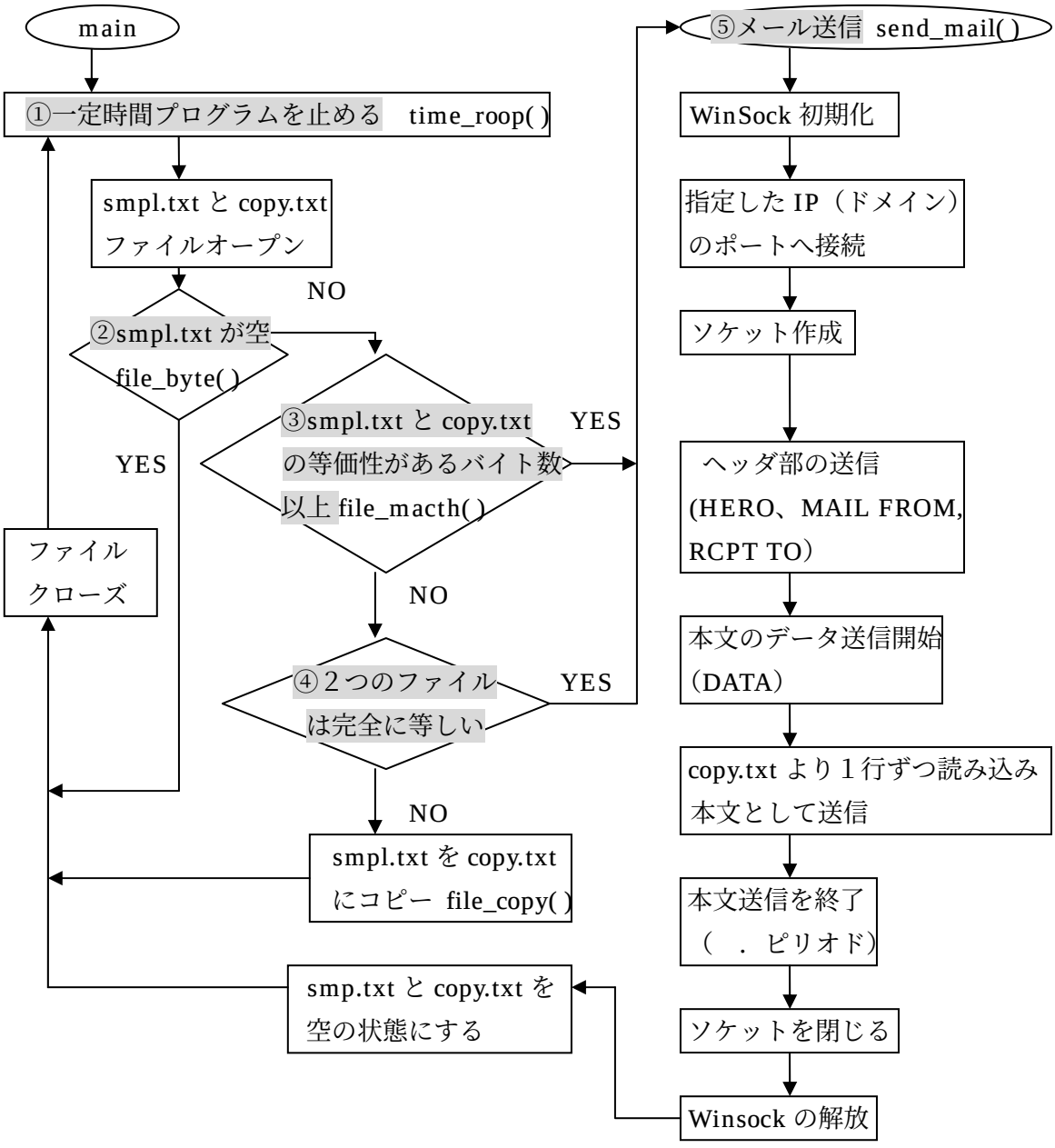
Winsock を使用したプログラムを制作する場合、Windows システムディレクトリに存在する wsock32.dll を使用する事となる。従って、プロジェクトの設定でライブラリファイル wsock32.lib を参照する必要があり、wsock32.dll は [ビルド]-[設定]-[リンク]-[一般]-[オブジェクト/ライブラリ モジュール] に wsock32.lib を追加することで使用可能となる。

4.2.4　メール送信プログラム

●フローチャート

・smpl.txt：snort の改良によってメールで送信するメッセージが書き出されるリアルタイム更新ファイル

・copy.txt：ある一定の周期ごとに更新される smpl.txt のコピーファイル



●主な関数説明

- time_roop( ) ：time 関数を使い、プログラムの進行を止める関数。
- file_byte( ) ：ファイルのバイト数を返す関数。
- file_macth( ) ：2 つのファイルを比較し、一致した部分のバイト数を返す関数。
- file_copy( ) ：ファイルをコピーする関数。
- send_mail( ) ：WinSock を用いたメール送信プログラム。

●概要

①メール送信プログラムの進行を一定時間止める。

メール用メッセージがどれだけ書き出されるかわからないので、一定間隔で確認する。欠点として不正アクセス検知からメール送信までに多少タイムラグが出てしまう。テストでは 10 秒間隔となっている。

②分岐：Smpl.txt が空の状態

snort によって書き出される smpl.txt が空の状態であるならば、snort による検知が行われていない、つまり不正アクセスはなかったという事とである。

③分岐：Smpl.txt と Copy.txt を比べて等価性があるバイト数以上

メッセージが書き込まれ続けるのを防ぐ処置。

テストでは 5000 バイトとしてある。

④分岐：Smpl.txt と Copy.txt を比べて２つのファイルが完全に等しい。

テストでは Copy.txt は新しい smpl.txt の 10 秒前のファイルである。その２つのファイルが完全に等しいということは snort による検知が一区切りしたと判断できる。

⑤メール送信

「4.2.2 メール送信手順」、「4.2.3 WinSock」で述べた流れに沿ってプログラムを組んである。テストでは SMTP サーバを 10.210.40.7、メール宛先を s99413@cis.shimane-u.ac.jp としている。

５章　テスト・実行結果

5.1　nmapNT

引用 web[４]

動作テストは「nmapNT」というポートスキャンツールを使用して行った。
「nmapNT」とは、linux 環境で動作する「nmap」と呼ばれるポートスキャンを高速に行うツールを Windows に移植したツールで、ターゲットのコンピュータの開いているポートを調べることができる。実行結果から、不要なポートが開いていないか、セキュリティ上問題のあるサービスが立ち上がってないか、また、管理者の知らないサービスが立ち上がってないかなどが確認できる。

コマンド書式

->　nmapnt　スキャンタイプ　オプション　ホスト名もしくネットワーク名

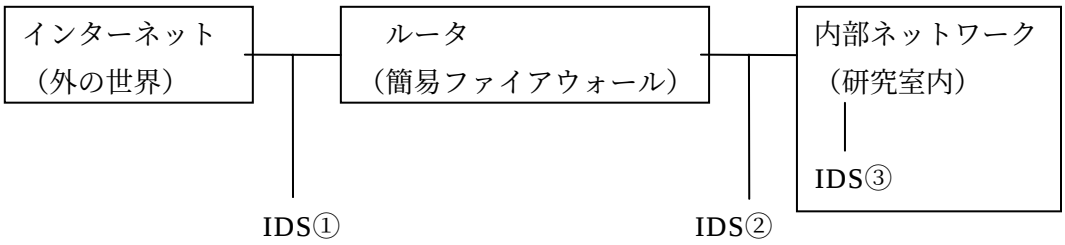
以下に代表的なスキャンタイプ、オプションを挙げる。

[スキャンタイプ]
　-sT：「スタンダード」を意味し、スキャンする各ポートに対して通常の TCP の通信コネクションを確立するか否かをチェックする。
　-sU：UDP を使ったサービスについてスキャンする。
　-sP：ネットワークにどれだけ動いているコンピュータが存在するか探すためのスキャン方法。

[オプション]
　-p：ポート番号の指定。「-p 20-30,139,60000-」と指定した場合、ポート番号 20～30 と 139 それと 60000 以上のポートをスキャンする。
　-g：こちらが使うポート番号を指定できる。
　-P0：スキャンするときに ping を発信して相手サーバが動いているかどうか調査しないというもの。一部のサイトはファイアウォールで ping に応答しないように設定してあったりするので、ping に応答しない＝サーバはダウンしている、というようなに間違ったスキャン結果を防ぐ。
　-PI：ping を使うと明示的に示すときに使用。（デフォルトで ping は使っている）

5.2 テスト

IDS の設置方法としては以下の３種類がある。



IDS①

この配置ではファイアウォールに対する攻撃の試みを検知することができる。

IDS②

この配置では IDS はファイアウォールを通り抜けることのできた攻撃を検知する。

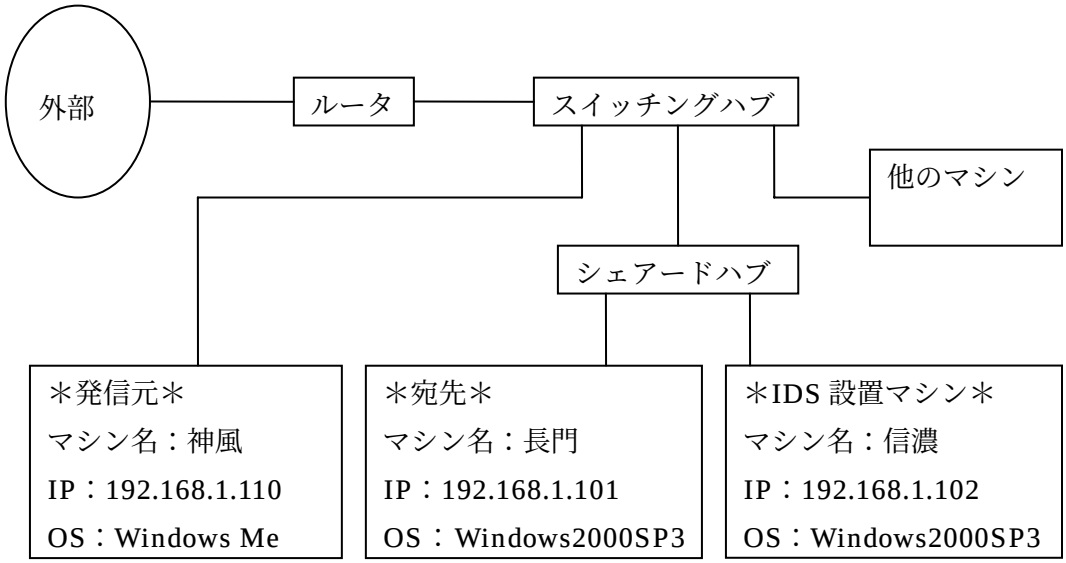
IDS③

内部のネットワークに IDS を配置することで内部からの攻撃を検知することができる。

テストはファイアウォールを破る攻撃は困難、また外部マシンに迷惑をかけないという理由より「IDS③」の配置で行った。

●研究室内の配置

テストを行うための研究室内の各マシン配置は次の通りである。



この配置はシェアードハブ（別名：リピータハブ）の受信したデータを他の端末すべてに送信するという特徴を利用して構成した。それにより、この構成で IDS が検知を試みるのは以下の４パターンとなる。

① マシン名：長門に対する通信
② マシン名：長門からネットワーク内（192.168.1.0/24）への通信
③ マシン名：信濃に対する通信
④ マシン名：信濃からネットワーク内（192.168.1.0/24）への通信

テストを行うにあたっての攻撃の方法は Windows 用ポートスキャンツール「nmapNT」を使って以下のコマンドを実行する。コマンドの意味は、発信元から宛先へ１番ポートから８０番ポートまで SYN パケット（通信開始要求）を送り、コネクション確立に対して相手からの返信が許可（SYN/ACK パケット）、拒否（RST/ACK パケット）でポートが開いているかどうかを判断するというもの。

```
->  nmapnt –sT –p 1-80 192.168.1.101
```

5.3　実行結果

表 5.3.1 は神風（発信元）が nmapNT を用いて長門（宛先）の１番から８０番ポートにスキャンをかけたときに出力された結果である。これを見ると長門の開いているポートが一目でわかる。

```
C:¥Nmap>nmapnt -sT -p 1-80 192.168.1.101



Starting nmapNT V. 2.53 SP1 by ryan@eEye.com
eEye Digital Security ( http://www.eEye.com )
based on nmap by fyodor@insecure.org  ( www.insecure.org/nmap/ )

Interesting ports on 長門 (192.168.1.101):
(The 70 ports scanned but not shown below are in state: closed)
Port        State       Service
7/tcp       open        echo
9/tcp       open        discard
13/tcp      open        daytime
17/tcp      open        qotd
19/tcp      open        chargen
21/tcp      open        ftp
25/tcp      open        smtp
42/tcp      open        nameserver
53/tcp      open        domain
80/tcp      open        http

Nmap run completed -- 1 IP address (1 host up) scanned in 8 seconds
```

表 5.3.1　nmapNT 時の神風のコマンドライン

以下の表5.3.2はメール送信プログラムにコメント出力を加えて実行したときの結果である。網掛け部が説明となっている。

```
0 秒経過
sec: wait...end
10 秒経過
sec: wait...end
20 秒経過
sec: wait...end                    ←ポートスキャン開始
ファイルコピー
smpl.txt=397byte
smpl.txt と copy.txt の一致=0byte
30 秒経過
sec: wait...end
ファイルコピー
smpl.txt=676byte
smpl.txt と copy.txt の一致=397byte
40 秒経過
sec: wait...end
smpl.txt=676byte
smpl.txt と copy.txt の一致=676byte                    ←メール送信開始
220 ad007.cis.shimane-u.ac.jp ESMTP Sendmail 8.9.3/3.7Wpl2/01040701; Tue,
11 Feb
 2003 06:01:34 +0900 (JST)
250 ad007.cis.shimane-u.ac.jp Hello jive.cis.shimane-u.ac.jp [10.210.8.12], plea
sed to meet you
250 shinano... Sender ok
250 s99413@cis.shimane-u.ac.jp... Recipient ok
354 Enter mail, end with "." on a line by itself
250 GAA06876 Message accepted for delivery
221 ad007.cis.shimane-u.ac.jp closing connection    ←メールサーバとの通信
区切りと見なし、メール送信
50 秒経過
sec: wait...end
```

表 5.3.2　メール送信プログラム実行時のコメント

図 5.3.3 は実際にメールとして届いたメッセージ、表 5.3.4 はそのときの信濃に残ったアラートである。きちんとメールは送られている。

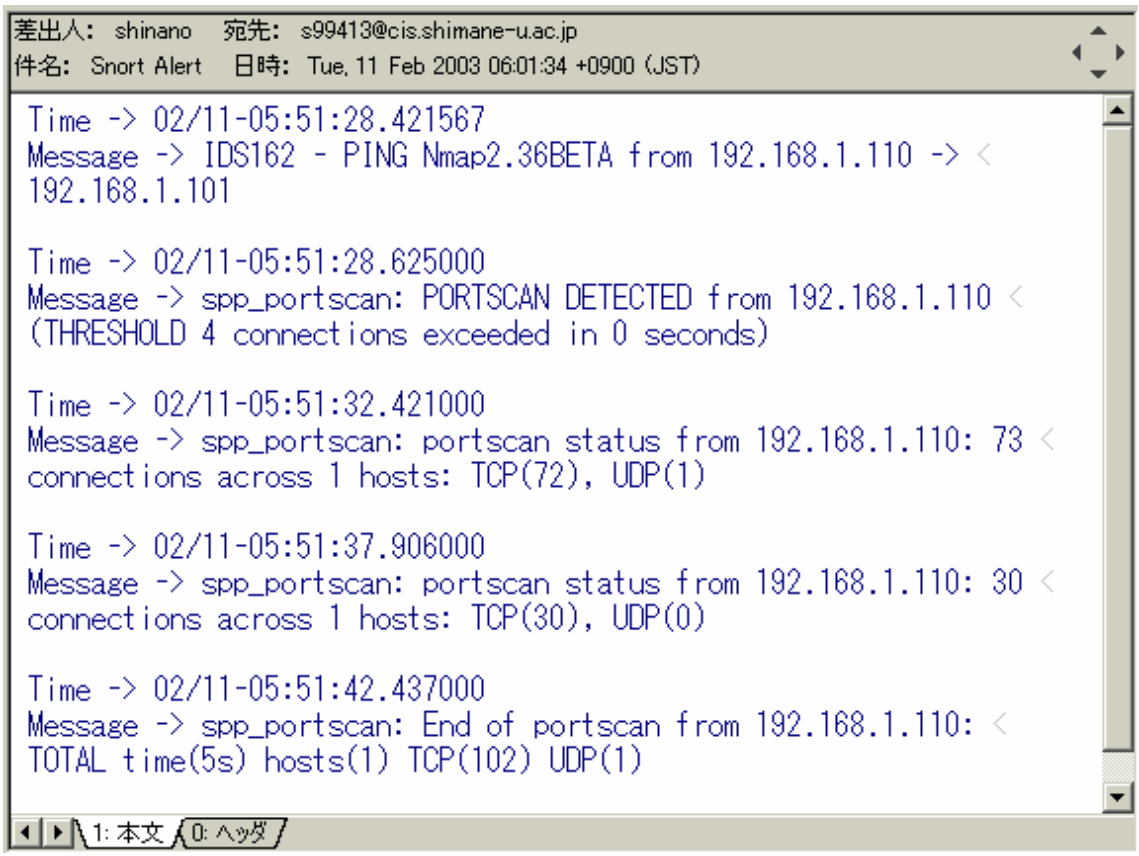


図 5.3.3　届いたメール（メールソフト：Becky!Ver.2）

```
 [**] IDS162 - PING Nmap2.36BETA [**]
02/11-05:51:28.421567 192.168.1.110 -> 192.168.1.101
ICMP TTL:41 TOS:0x0 ID:48451 IpLen:20 DgmLen:28
Type:8  Code:0  ID:48390   Seq:0  ECHO

[**] spp_portscan: PORTSCAN DETECTED from 192.168.1.110 (THRESHOLD 4 connections exceeded in 0 seconds) [**]
02/11-05:51:28.625000
[**] spp_portscan: portscan status from 192.168.1.110: 73 connections across 1 hosts: TCP(72), UDP(1) [**]
02/11-05:51:32.421000
[**] spp_portscan: portscan status from 192.168.1.110: 30 connections across 1 hosts: TCP(30), UDP(0) [**]
02/11-05:51:37.906000
[**] spp_portscan: End of portscan from 192.168.1.110: TOTAL time(5s) hosts(1) TCP(102) UDP(1) [**]
02/11-05:51:42.437000
```

表 5.3.4　信濃（192.168.1.102）に出力されたアラート

これを見ると Nmap によるポートスキャンがルールに反応して検知されていることがわかる。ただし、どれだけの時間内に何回接続があったはわかるが、どのポートがスキャンされたかはわからない。

今回参照されたルールは ping-lib ファイル内にある以下の部分。

①　②　③　④　⑤　⑥

```
alert icmp any any -> $HOME_NET any (msg:"IDS162 - PING Nmap2.36BETA";itype:8;dsize:0;)
```

⑦　⑧

[意味]

3　ルールヘッダ部

① アクション：alert　ルールオプション内の指示に従いアラートを出力

② 検知すべきプロトコル：ICMP

③ 発信元 IP、ポート番号：ともにすべて

④ 検知すべきデータの流れ：発信元から宛先へ送られるデータ

⑤ 宛先 IP、ポート番号：$HOME_NET（192.168.1.0 /24 と設定済み）、ポート番号はすべて

4　ルールオプション部

⑥ ルールにマッチした場合は「IDS162 - PING Nmap2.36BETA」というメッセージを出力

⑦ ICMP の type フィールドの値を評価

⑧ パケットペイロード（通信文）のサイズの評価

メールでは⑦、⑧といった部分がわからないが、アラートを見ればこの部分もルールとマッチングしているということがわかる。

IDS に関する情報は以下のサイトで詳しく解説されているので、そちらを参照にしていただきたい。

http://whitehats.com/ids/index.html

6章　まとめ

6.1　考察

最新の攻撃・侵入手法については十分な情報が無いことから、これらの攻撃・侵入に対するルールを記述できない場合がある。この期間に最新の攻撃・侵入手法を用いられると snort は未知の攻撃として検出することができない。新たな攻撃手法やセキュリティホールは日々発見されており、これに対応してネットワークも常にメンテナンスし続けていかなければならない。

この欠点の克服には以下の例のようなルール追加で検出可能であると考えられる。

例１　特定のポート以外をすべて記録する

```
pass tcp any any < > 192.168.1.0/24 25
pass tcp any any < > 192.168.1.0/24 80
alert tcp any any < > 192.168.1.0/24 any(msg:”tracking”)
alert udp any any < > 192.168.1.0/24 any(msg:”tracking”)
alert icmp any any < > 192.168.1.0/24 any(msg:”tracking”)
```

SMTP（25 番）や HTTP（80 番）といったようなよく使われるポートのトラフィックを完全に無視して、他のトラフィックを記録するという方法。

例２　特定のポートに対する通信をキャプチャする

```
alert udp any any < > 192.168.1.0/24 1434 (msg:”traffic to 1434/udp”);
```

1434 / UDP は通常 SQL Server の解決サービスのポートであるが、最近話題となった SQLSlammer よる攻撃データの入り口でもある。

（SQLSlammer について参照は
http://www.nai.com/japan/virusinfo/virS.asp?v=W32/SQLSlammer.worm）

このようなルールを作成してやれば未知の攻撃に対してもログをとることができるはずである。しかし、そのポートで通信が行われただけでルールとマッチングしてしまい、snort が攻撃の可能性があると判断し、たとえ本当に攻撃があった場合ではなくても警報を発してしまい誤検知である可能性が高くなってしまう。

snort が稼動しているからといって、そのネットワークが防御されているわけでは無い。snort が発したアラートが誤報であるか否かの判断は必ず管理者が行い、それに従った正しい対応を行うべきである。

6.2　今後の展望

侵入検査システムにより不正アクセスを検知して早期に管理者に知らせる事はできるのだが、セキュリティの強化をするまで不正アクセスとわかっていても通信を拒否することはできない。そのため不正アクセスの検知からセキュリティを強化するまで、被害の軽減のため一定時間だけアクセスを遮断、パケットを破棄するような機能が必要である。

また、そのような機能を IP を調べてネットワーク内の各マシンで動かせるようにするエージェント化も重要となる。

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先生には、心より御礼申し上げます。また、田中研究室の貫目洋一さん、辰己圭介さん、高木明さん、喜代吉容大君、鷲見明君、垰田千帆さん、田辺知里さん、堀隆志君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。

引用文献

[1] オーム社、「不正アクセス調査ガイド」
渡辺勝弘＋伊原秀明＝著、314pp、2002

[2] オーム社、「基礎からわかる TCP/IP ネットワーク活用ツール」
井口信和＝著、319pp、2000

[3] （株）ピアソン、「インターネットプログラミング」
コア・ダンプ＝著、265pp、1998

[4] office・みっきー、「無料ツールで作るセキュアな環境」
https://www.netsecurity.ne.jp/article/3/2405.html 、2001

ツールや説明等の Web

snort
http://www.snort.org/

WinPcap
http://netgroup-serv.polito.it/winpcap/

snort.panel
http://www.xato.net/files.htm

nmapNT
http://www.eeye.com/html/Research/Tools/nmapNT.htmlwo

SQLSlammer について
http://www.nai.com/japan/virusinfo/virS.asp?v=W32/SQLSlammer.worm

IDS に関する情報 Whitehats
http://whitehats.com/ids/index.html