修士論文 2001年度 (平成13年度)

統一的な
セキュリティポリシー管理機構の実現

慶應義塾大学 大学院 政策・メディア研究科
有賀 征爾

ariga@sfc.keio.ac.jp

修士論文要旨 2001 年度 (平成 13 年度)

統一的なセキュリティポリシー管理機構の実現

論文要旨

本研究では，大規模ネットワークにおいてパケットフィルタリングを行なう際のセキュリティポリシーを統一的に管理するためのシステムを提案し，安全なネットワークを容易に構築・運用するという目的を実現した．

インターネットの利用者が増加し，インターネットが一般の重要な通信インフラとして利用されるようになった．企業や家庭にも広く普及し，インターネットを通じた商取引も，企業間・企業個人間を問わず，活発に行なわれている．その一方，インターネットプロトコルが，セキュリティの機能を持っていないことに乗じて，さまざまな攻撃がインターネット上で行なわれるようになった．攻撃の手法の多様化から，それに対する防御策も多岐に渡っている．

攻撃に対する防御策として，現在最も一般的に使われているのは，パケットフィルタリング技術である．パケットフィルタリング技術は，ファイアウォールや IPsec などに使われる基礎的な技術であり，インターネットにおける基本的な安全性を提供するものである．しかし，大規模ネットワークにおいて複数のパケットフィルタリング機構を運用する上で，パケットフィルタリングルールの非統一，セキュリティポリシーの配布機構の欠落などの，統一的なセキュリティポリシー管理機構の欠如が問題となっている．

本機構は，これらの問題を解決するため，汎用的なセキュリティポリシー記述言語，セキュリティポリシーリポジトリ，リポジトリとパケットフィルタリング機構間においてセキュリティポリシーの配布を行なうポリシー配布エージェント，そして各パケットフィルタリング機構からの情報を収集する機構のそれぞれを提案し，統一的なセキュリティポリシー管理機構として実現した．これによってフィルタリングネットワークの管理をより安全に容易にするという目的を達成した．

キーワード

1. インターネット，2. セキュリティ，3. パケットフィルタリング，4. ポリシー管理，
5. IPsec，6. ファイアウォール

慶應義塾大学 大学院 政策・メディア研究科
有賀 征爾

Abstract of Master's Thesis Academic Year 2000

Integrated Network Security Policy Management System

Summary

This research proposes the architecture to control the security policy integratively when filtering the packets at large networks, and achieved to easily develop and operate a secure network.

The number of the Internet users has grown rapidly and the Internet is used as one of the most important communication infrastructure publically. The Internet is popularized through enterprises and home widely and commercial transactions using the Internet is carried out actively in both BtoB and BtoC areas.

On the other hand, various attacks are done on the Internet, taking advantage of the internet protocols having no security feature from its scheme. Details of the attacks vary, and the hedges against the attacks branch out.

The most common technique used for the defense against the attacks is packet-filtering. Packet-filtering is a basic technology used for firewall and IPsec, which provides elemental security on the Internet.

However, the absence of the integrated security-policy control machanism such as unstructured packet-filtering rules and the lack of the security-policy distribution mechanism has been the problem when operating multiple packet-filtering mechanism on large networks.

The mechanism proposed in this paper is implemented to solve the problems mentioned above by providing integrated security-policy control mechanism, which consists of well-understood security-policy description language, repository for security-policy, policy-distribution agent which distributes the security-policies between repository and packet-filtering mechanism, and mechanism to collect information from each packet-filtering mechanism.

Keyword

1. Internet, 2. Security, 3. Packet filtering, 4. Policy Management, 5. IPsec, 6. Firewall

Keio University Graduate School of Media and Governance
Seiji Ariga

目 次

図 目 次

表 目 次

第1章　序論

1.1　はじめに

ここ数年で，我々をとりまくインターネット環境は劇的に変化した．インターネットが一般の通信インフラとして日常的に使われるようになり，さらにはその役割を他の通信インフラの補助的なものから，通信インフラの中心的なものへと変えてきた．

企業での利用方法を見ると，社内での情報のやり取りの多くに電子メールや知識共有データベース (多くの場合，いわゆる電子掲示版の形態をとっている) などが使われるようになった．また社内だけでなく外部への情報提供には WWW が多く使われ，企業間取り引きにも積極的にインターネットが使われるようになった.「平成１２年度電子商取引に関する市場規模・実態調査」 [2] によると，2000 年の企業間の電子取り引きの市場規模は約 22 兆円，企業個人間の電子商取り引きは約 8240 億円となっており，それぞれ前年に比して 60%と 145%という急激な成長を遂げた．

また，家庭内にも広くインターネットが普及しつつある．オンラインショッピングが日常的に行なわれるようになり，銀行の残高の照会，振り込みや株の取り引きなどもインターネットを経由して行なうことができるようになった．

しかしその一方で，この新しい社会インフラに対する脅威も非常に高まっている．

たとえば，WWW サーバを初めとする，インターネット上のサーバに対する侵入事件，またサーバに侵入されることによって起こるコンテンツの改ざん事件が多発した．Attrition.org が報告した WWW 被害の統計 citeattrition によると，2001 年の第一四半期には，一日約 70 件程度の WWW 改ざん事件が発生している．

またコンテンツの改ざんだけではなく，企業の機密情報や顧客の個人情報の流出なども現実に発生した．さらに，情報の改ざんや流出だけでなく，サービス妨害攻撃 (DoS) のように，あるサーバがサービスの提供を困難にするような攻撃形態も一般的になった．

1.2　インターネット上の脆弱性の要因

これらの問題は，単一の要因によるものではなく，それぞれ異なる要因，またはそれらの組み合わせによっておこる．インターネット上における脆弱性の要因には，たとえば次のようなものがある．

ソフトウェアの欠陥

インターネット上の安全性の脅威の要因で一番多いのがソフトウェアの欠陥である．上にあげたウィルスやサーバへの侵入などの多くはソフトウェアの欠陥を利用して，アプリケーションやオペレーティングシステムの予期しない状態を引き起こし，悪意のあるプログラムを実行させる．

単純に欠陥と言っても非常に多くのものがある．しかし，どの欠陥にも言えることは，基本的には問題のあるソフトウェアの問題の個所だけを修正すればよいということである．

利用者・管理者のミス

インターネットが重要な通信手段として使われるようになるに従って，利用方法も簡単になっていった．その結果，利用者層が広がり，プロトコルなどをまったく知らない利用者でも日常的に使えるようになった．しかしそのため，たとえばメーリングリスト・メールマガジンの管理者が購読者の個人の情報をメーリングリストに流してしまうといった，利用上のミスも度々起こるようになった．

もちろんインターネットの初期にもそのような問題はあったと思われるが，現在のように広く一般に使われるようになると，一度のミスによる影響ははるかに大きくなっており，またその頻度も増加した．

また，ソフトウェアを使う上では，さまざまな設定が必要となるが，システムに精通していない管理者が設定したことによって，そのシステムが脆弱なものになってしまうこともある．

さらに，上に挙げたソフトウェアの欠陥が見つかった場合，そのソフトウェアを更新し，欠陥の修正という作業が必要となる．通常，この作業は利用者．管理者が行なうが，多くの利用者・管理者がその欠陥に気がつかず，脆弱性を放置してしまう危険性がある．そのため，ソフトウェアの開発元が脆弱性を修正しても，実際の利用者のもとでは放置されてしまい，その脆弱性を利用した攻撃が継続して可能となる．

これらの問題の解決には，もちろん利用者・管理者の啓蒙・学習といったことも必要である．しかし，ソフトウェアの自動更新や柔軟なアクセス制限などにより，システムとして安全性を保つ仕組みも必要であり，実際にさまざまなシステムが考えられた．

アーキテクチャ上の欠陥

あるシステムやプロトコルのアーキテクチャから発生する脆弱性もある．

たとえばインターネット上の最も重要なシステムの一つである，DNS (Domain Name System) は，あるクエリーに対する応答に対する認証を行なっていないため，悪意のある第三者があるクエリーに対して偽の応答を返すことができる．クエリーの送信者は応答が偽りのものと認識できないため，意図していないホストと通信するという事態が発生してしまう．

これは，DNS というシステム・プロトコル自体がクエリーとその応答の認証を必要としないというプロトコル上の欠陥であり，ソフトウェアの欠陥ではない．[1]

この問題はソフトウェアの欠陥と違い，問題となったソフトウェアの欠陥だけを直せばいいという問題ではなく，あるプロトコル (やアーキテクチャ) に問題が発見された場合，そのプロトコルに従って実装されたソフトウェアをすべて修正しなければならない．

アーキテクチャの仕様外の問題

これは，あるシステムやプロトコルが作られたときには想定していなかった事態が起こることによる問題である．

インターネットの初期には通信の安全性は特別に重要な問題ではなかった．これは研究ネットワークとして広く用いられていたという背景によるところが多く，通信はすべて平文で行なわれていた．

当時は「通信ができる」ということが最も重要であり，そこで通信されるデータも機密性の低いものであったため，この仕様は妥当であったといえる．

[1] この問題はすでに認識されており，応答の認証や通信路の暗号化を行なうという方向で仕様の策定が進んでいる．

しかし，現在のようにインターネットが社会の重要なインフラとして使われるようになると，このような仕様が実際上の問題として立ち現れてくる．

たとえば電子メールはインターネット上で最もよく使われるアプリケーションの一つであり，そのプロトコルの策定もインターネットの初期に行なわれた．そのため，電子メールは，その配送経路上で特に暗号化されることなく，そのままの形で通信される．そのため配送経路上の悪意のある第三者はその中身を見ることができてしまう．したがって機密情報を電子メールでやりとりすると，情報漏洩の経路の一つとなってしまう可能性が非常に高い．

この問題も，アーキテクチャ上の欠陥と同じように，プロトコルを改良し，現行の実装をすべて修正するか，もしくは，同じ機能を持つまったく新しいプロトコルを作成し，現行のシステムをそちらに移行するといったことでしか対処できない．

このようにインターネット上における脅威の要因はさまざまであり，それぞれに個別に対処するしかない．しかし，時間も資源も有限であり，すべての問題に個別に対処するのは事実上不可能である．そこで多くの場合，次のような対処を行い，問題を集約する．

アクセスコントロール

すべてのノードのすべてのアプリケーションの安全を保つのは非常にコストがかかるため，自ネットワークの外部からのアクセスを制限するのが一般的である．外部からのアクセスが必要なノード，またそのアプリケーションを把握し，その他のノードやそのアプリケーションにはアクセスできなくしてしまう．このようにすることによって必要なアプリケーション・サービスのみの安全の確保と，アクセス制限の管理だけをすればよいことになる．

このアクセスコントロールは通常，ファイアウォールを使って行なわる．小規模なネットワークの場合は，自ネットワークを一つのネットワークとして扱い，その出入り口 (ゲートウェイ) でのみアクセス制限を行なう．大規模なネットワークでは，自ネットワーク複数の小規模なネットワークに分割し，それぞれのネットワークが連携して必要なアクセス制限を行なう．

アクセスコントロールは通信の始点・終点や，利用するプロトコル・アプリケーションに基づいて行なう．

下位レイヤでの暗号化と認証

インターネットの初期に策定され，現在に渡っても使われているシステム・プロトコルは，先に説明したように安全な通信を実現するための機能を持っていないことがほとんどである．そこで，ネットワーク層やデータリンク層で暗号化や認証を実現し，上位層である，TCP/UDP 層やアプリケーション層のプロトコルに改変を加えることなく安全な通信を実現する方法が，広く使われる．

これは主に VPN(Virtual Private Network) というかたちで実現されており，実際のプロトコルには IPsec(IP Security) や L2TP(Layer2 Tunneling Protocol) などが使用される．

ここで分かることは，どちらの方法でもアプリケーション・プロトコルに依存しないように下位のレイヤで安全を確保していることだ．通信をゲートウェイにおいてパケット単位で処理し，そのパケットの属性 (始点・終点・ペイロードの内容) にしたがって，処理の内容を決定する．ルータ (ゲートウェイ) においてパケットをただ転送するのではなく，パケットの属性にしたがってなんらかの処理を行なうことを，パケットフィルタリングと呼ぶ．

このように，現在のインターネットは全体として見ると，さまざまな場所でパケットフィルタリングが行なわれる．さらに大規模なネットワークではこれらのパケットフィルタリングを，自ネットワーク内の複数の地点で行なっている．この複数のパケットフィルタリングは独立したものではなく，ネットワークの安全を確保するという目的のために，連携して動作しなければならない．

したがって大規模なネットワークの管理者は，自ネットワークの複数の地点で行なわれるパケットフィルタリングを統一的に扱い，あるゲートウェイで行なった変更が，他のゲートウェイでの状態と衝突していないことを逐一確認する必要がある．

これは，ネットワークの規模が大きくなればなるほど困難となり，ゲートウェイの管理を集中的に行なうことのできるシステムが必要とされる．

1.3 本論文の目的

本論文では，大規模ネットワークにおけるパケットフィルタリングの管理・運用を容易にし，安全なネットワークの構築を支援することを目的とする．

本研究では現状の問題点を解決するために，パケットフィルタリングルールの統一的フォーマットを規定し，パケットフィルタリングを行なうゲートウェイからのフィルタリングルールの収集，フィルタリングルールの管理，そして再度，ゲートウェイへのフィルタリングルールの配布を行なうことによって，大規模ネットワークにおけるパケットフィルタリングを，統一的に管理するためのシステムを提案する．

これによりネットワークの管理者は一意性をもったフィルタリングポリシーを，ネットワーク全体で容易に実現することができるようになる．

1.4 本論文の構成

第 2 章では，本論文が対象とするパケットフィルタリング技術の定義と，実際に利用される，いくつかのパケットフィルタリング技術の実装について説明する．第 3 章では，現状のパケットフィルタリング技術を大規模ネットワークで管理・運用する際に起こる問題を説明する．

第 4 章では，第 3 章で明らかにした問題点を解決するための手法を提案し，第 5 章で，これらの手法を適用した統一的なパケットフィルタリング管理機構を実現，既存の機構との比較評価を行なう．

第 6 章では，まとめとして本論文の結論と今後の課題について述べる．

第2章　フィルタリング技術とセキュリティポリシー

本章では，初めに現在のネットワークセキュリティの現状について説明する．

次に，対策の中でも最も広く行なわれているパケットフィルタリングについて，現在の実装に基づいて説明し，本研究が対象とするパケットフィルタリングの定義を行なう．

最後に，セキュリティ対策を行なう上で，ネットワークの設計・構築・運用の基礎となるセキュリティポリシーの説明を行なう．ここで，広範な意味を持つセキュリティポリシーの中で，本研究の対象とする領域を定義する．

2.1　ネットワークセキュリティの必要性

1章で述べたように，もともとインターネットでは通信の安全性が提供されない．したがって，安全な通信を行ないたい場合は，利用者自身が安全な通信環境を別に構築する必要がある．

本研究では「通信環境の安全性」を，能動的な安全性と受動的な安全性の二つに分けて考える．能動的な通信環境の安全性とは次の場合を指す．

- 利用者が通信を開始するときに必要とされる安全性．リモートログインをする場合や，機密性の高いメールなどを送信する場合，WWW でクレジットカード番号を送信する場合などが該当する．

受動的な通信環境の安全性とは次の場合を指す．

- 自分の資源を攻撃者から守るための安全性．WWW でサービスを提供する場合，電子メールを読む場合など，一般にコンピュータをインターネットに接続する場合に，常に必要となる安全性である．

この二種類の安全性の違いを表 2.1に簡単にまとめる．

表 2.1: 能動的な安全性と受動的な安全性

	必要とされる時	対象	必要とする人
能動的な安全性	通信時に必要	攻撃から通信を守る	主にエンドユーザ
受動的な安全性	常時必要	攻撃から資源を守る	主に管理者

以下，安全な通信を，この二種類の安全性に分けて述べる．

2.1.1 能動的な安全性

能動的な安全性とは，利用者が能動的に通信を開始するときに必要となるものである．たとえインターネットにつながっていてもまったく通信を行なっていない場合，つまりはユーザが意図的に通信を行なっていない場合は，この安全性の対象とはならない．また，ここで述べる安全な通信とは，通信の機密性・認証・完全性などの機能を，利用者が必要に応じて利用できることを指す．

今日のインターネットにおいては，能動的な安全性を確保するためには，特別に安全な通信路を提供するアプリケーションやプロトコルを利用する必要がある．能動的な安全性を提供するアプリケーションやプロトコルはすでにさまざまなものが存在する．以下に簡単に列挙する．

- TLS [3]
- PGP [4], S/MIME [5]
- SSH [6]
- 安全な認証サービス (CHAP，Digest 認証，Kerberos)

ここに挙げたものは，どれも基本的にはトランスポート層，もしくはセッション層以上のプロトコル，アプリケーションプロトコルであり，これらの機能を利用するためにはアプリケーションの改変が必要となる．したがって，利用者が安全な通信を実現しようと考えても，利用するアプリケーションがこれらのプロトコルに対応していない場合，安全な通信環境を実現することができない．

たとえば，電子メールをメールサーバから取得するプロトコルとして Post Office Protocol Version 3 (POP3) [7] があるが，このプロトコルでは通信内容は平文のままででやりとりされる．したがって，通信系路上で悪意のある第三者が通信を盗聴していれば，その内容は筒抜けになってしまう．

そこで，POP3 over TLS [8] のように POP3 を TLS によって暗号化し，通信内容を秘匿するという方法も提案された．しかし，アプリケーション (電子メールクライアント) とサーバ (メールサーバ) の双方がこのプロトコルに対応していないければ，このプロトコルを使うことはできない．実際に，このプロトコルに対応した電子メールクライアントは少なく，多くの POP3 の通信は平文のまま行なわれるのが現実である．

そこで，トランスポート層より下位の層である IP 層で安全な通信を実現するための機構が実現された．下位の層で安全な通信を実現することにより，上位層のプロトコルやアプリケーションは意識することなく安全な通信を行なうことができる．これによって利用者は，アプリケーションを改変することなく安全な通信を行なえる．

このようなプロトコルとして，現在 IP Security(IPsec) [9] が広く使われる．IPsec については 2.2.2節で詳しく述べる．

2.1.2 受動的な安全性

受動的な安全性とは，利用者が通信を行なっていない場合でも必要となる安全性のことである．インターネットに接続するコンピュータは常に外部からの攻撃の危険性に晒される．現在頻発する WWW サーバへの侵入などはこの種の危険性の一つである．まだドラフトではあるが情報処理振興事業協会の報告書 [10, pp. III-1] によると，この数年 WWW サービスを狙った攻撃が非常に多い．WWW サービスなどをインターネットで提供する場合，この受動的な安全性の確保が重要となる．

またコンピュータウィルスも，この受動的な安全性の対象となる．コンピュータウィルスは，今日では電子メールを通して感染・伝播することが多い．この場合も自分が情報を外に流すのではなく，電子メールという形をとって，外部から攻撃を受けるため，受動的な安全性の確保が必要となる．

受動的な安全性の確保は能動的な安全性の確保とは異なり，コンピュータ資源への適切なアクセス制限が重要となる．攻撃者は利用者の通信が開始されるのを待つのではなく，コンピュータへの不正アクセス，コンピュータウィルスの送付など，積極的な手段で利用者のコンピュータ資源の安全性を脅かす．

したがって，WWW サービスを行なうサーバでは，攻撃者によって不必要なプログラムにアクセスされないよう適切なアクセス制限をする必要がある．また，ウィルスに対してもウィルス対策ソフトを使って既知のコンピュータウィルスが重要なコンピュータ資源にアクセスしないよう，アクセス制限を行なう必要がある．

インターネットが社会の重要なインフラとして認知され，インターネット上のサービスが一般的になるに従って，広くこれらの対策が行なわれるようになった．

インターネット上における一般的なアクセス制限の手法としてファイアウォールがある．ファイアウォールは通常，ゲートウェイとして動作し受信したパケットの属性やその他の条件にしたがって，パケットを転送，または破棄といった動作を行なう．これによって意図しないホストからのアクセス，また意図しないホストやサービスへのアクセスを制限することができる．制限対象を適切に設定することによって，提供するサービスを信頼できるホストからのアクセスや，想定していないホストやサービスへのアクセスを制限することができる．

2.1.3 ネットワークセキュリティの現状

ここまでで述べたようにネットワークにおける安全性には，その性質によって二種類のものがある．現在のネットワークセキュリティはこの両方に対処することを目指す．

次に，これらのネットワークセキュリティのための手法を，ネットワーク上のどこで実現・実行するかが問題となる．

たとえば，ウィルス対策ソフトは長年エンドユーザの利用するコンピュータで実行されてきた．コンピュータウィルスは，攻撃者がコンピュータウィルスを送信してからネットワーク上を転送され，ユーザのメール受信ソフトに届いたときに初めてウィルス対策ソフトによってチェックされる．

これによって確かにコンピュータウィルスの脅威から，コンピュータ資源を守ることはできる．しかし，昨今のコンピュータウィルスの爆発的な増加，一般家庭へのコンピュータの普及により，徐々にこの図式が変わりつつある．つまりインターネットの端点であるエンドユーザのもとにおいて対策をするのではなく，その前，インターネットサービスプロバイダー (ISP) においてウィルス対策を行なうという例が増えた．

これはエンドユーザがウィルス対策をしなくてもよくなるという利点の他に，よりインターネットのバックボーンに近い，ISP においてウィルスチェックを行なった方が効率的にウィルス対策を行えるからである．

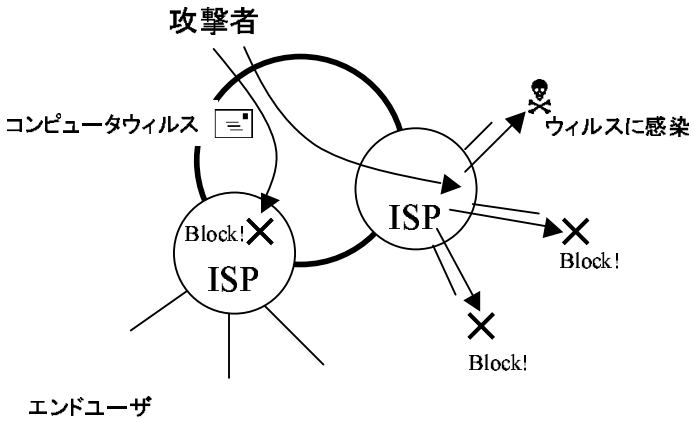

図 2.1: コンピュータウィルスの伝播

図 2.1に示すように，ISP がウィルスチェックを行なえば，チェックは一度で済むが，エンドユーザがウィルスチェックを行なうと，エンドユーザの個々のコンピュータでウィルスチェックを行なうことになり，ネットワーク全体で見たときのチェックの回数は非常に増大する．したがって，コンピュータウィルスのチェックは，ある組織単位のネットワークの入り口で行なったほうが効率的であると言える．

同じことは，IPsec やファイアウォールなどのシステムにも言える．IPsec を利用する場合，ホストごとに IPsec による通信を行なうこともできる．しかし，たとえば，地理的に離れた場所にある企業の事業所を IPsec によってインターネットを経由して接続する場合，各事業所のコンピュータすべてが IPsec を用いて通信するよりも，各事業所の出口であるゲートウェイ間において IPsec による通信を行なった方が管理コストを減少することができる．

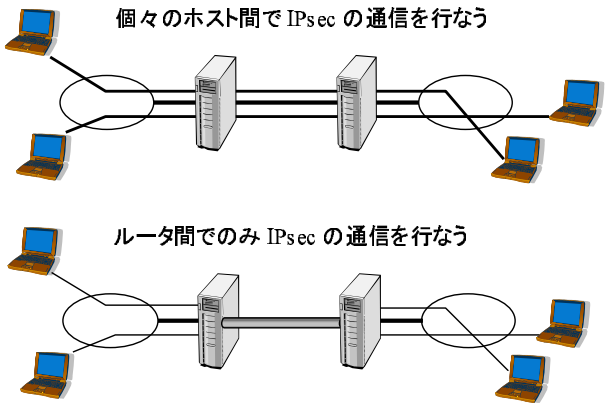

図 2.2: ホスト間とゲートウェイ間の IPsec

なぜなら IPsec に起因する通信障害が起こった場合，個々のコンピュータを調査するのではなく，ゲートウェイとなる二つのルータだけを調査すればよいからである．また，IPsec を利用した

ネットワークを設計する場合でも，各ホストそれぞれの通信を含めて設計するよりも，ゲートウェイ間の設計だけをすればよいので，管理者のコストは低減する．

また，ファイアウォールはその機能から，必ずバックボーンネットワークとエンドノードの間に位置することになる．もちろん昨今，パーソナルファイアウォールというかたちでエンドノードにおいてもファイアウォールの機能を果たす製品もあるが，ネットワークセキュリティの領域においては，ある自律ネットワークのゲートウェイでファイアウォールを運用する形態が主導である．

このように現在のネットワークセキュリティの中心的な手法は，エンドノードだけでなく通信の中間点となる，ある組織単位のネットワークのゲートウェイで行なわれることが多い．

ゲートウェイは通常，ルータとして動作する．このようにネットワークの中間点で，パケットの属性 (たとえば，ウィルスの添付の有無なども含む) によって通過させるかどうかを判断する機能を **パケットフィルタリング**と呼ぶ．パケットフィルタリングは一般にファイアウォールの機能として説明されることが多いが，本論文では広くネットワークの中間点でパケットの処理を行なうことを指す．

したがって，ファイアウォール と IPsec はパケットフィルタリングのシステムの一つであると言える．前者は受動的な安全性を実現し，後者は能動的な安全性を実現する．

次節 2.2で，パケットフィルタリングの概要，そしてファイアウォールの概要と実装，IPsec の概要と実装について述べる．

2.2 パケットフィルタリング技術の概要とその実装

先に述べたようにパケットフィルタリングとは，パケットの属性にしたがって以下のような処理を行なう．

- 破棄する
- 暗号化する
- そのまま転送する
- アドレス変換をする
- 記録をとった上で転送する

実際にどのような処理が行なわれるかは，パケットフィルタリングがどのような目的に使われるかによって異なる．

パケットフィルタリングは通常，IP 層において行なわれる．図 2.3にパケットフィルタリングの過程を示す．

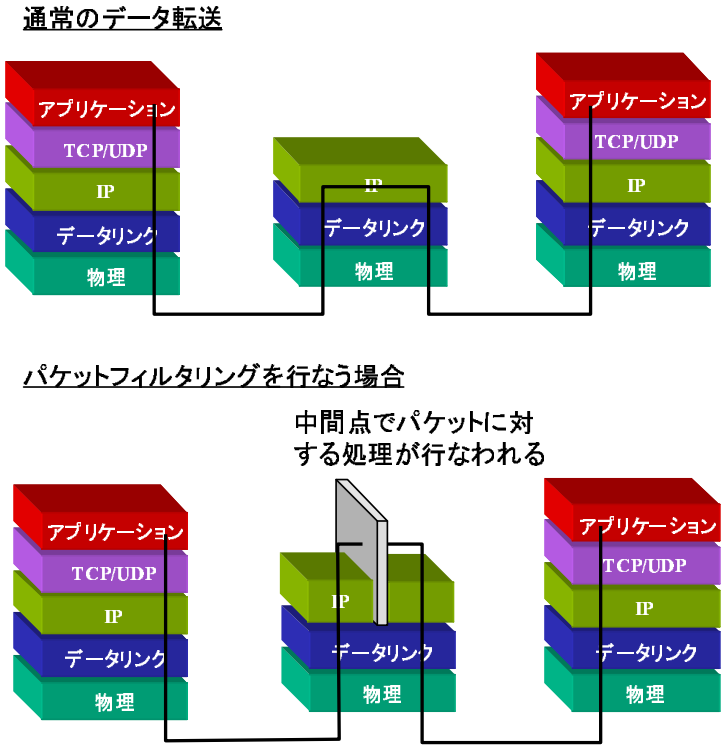

図 2.3: パケットフィルタリング

パケットはエンドノードから送信され，ルータ間を転送され受信者まで送られる．途中のゲートウェイで処理が行なわれ，たとえば処理内容が「そのまま転送する」である場合，パケットには何も改変が加えられず，受信者まで届く．送信者はもちろん，受信者もパケットフィルタリングが行なわれたことに気がつくことはない．

パケットフィルタリングは一般的なセキュリティ技術として知られており，多くのルータ，ファイアウォール製品で使われる．また，2.1.1 節で言及した IPsec でも多くの実装はパケットフィルタリングの手法を用いて IPsec の機能を実現した．

本節では現在広く使われる，パケットフィルタリング技術を用いた機構を，2.1節で述べた分類に基づいて説明する．

- 受動的な安全性を実現するファイアウォール
- 能動的な安全性を実現する IPsec

2.2.1 ファイアウォールにおけるパケットフィルタリング

本節ではパケットフィルタリングの代表的な機構であるファイアウォールを取り上げ，パケットフィルタリング技術が実際のシステムにおいてどのように利用されるかを述べる．

ファイアウォールの概要

ファイアウォールの目的は，ネットワークにおけるアクセス制限を行なうことである．アクセス制限はゲートウェイが転送するパケットの属性に基づいて行なわれる．使用されるパケットの属性には以下などが挙げられる．

- 送信元アドレス
- 送信先アドレス
- 利用しようとするサービス
- ICMP タイプ
- TCP フラグ

通常のパケットフィルタリングの実装では，パケットの受信時，パケットの送信時に，管理者によって設定されたパケットフィルタリングのルールに従ってパケットを処理する．

パケットはフィルタリングルールに従って，破棄，転送のどちらかの処理が行なわれる．管理者は特定のサービス・アドレスに対してのみ転送を許可し，その他のアドレスへのパケットをゲートウェイにおいて破棄することによって，管理下にあるネットワークへの外部からのアクセスを遮断する．

これは，内部から外部の通信があるかないかにかかわらず，常時行なわれるアクセス制限であり，したがって受動的な安全性を実現すると言える．

ファイアウォールのさまざまな実装

ここでいくつかの実装におけるパケットフィルタリングのルールを例示する．

前提とするネットワーク例は図 2.4の通りである．

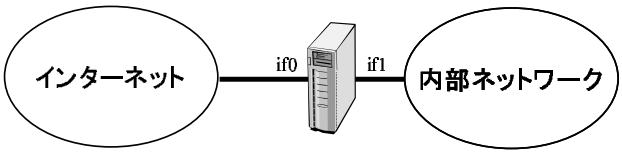

図 2.4: パケットフィルタリングネットワーク例

図中の「if0」「if1」はそれぞれゲートウェイのネットワークインターフェース名である．また，ゲートウェイにおけるパケットフィルタリングルールは表 2.2の通りであるものとする．

表 2.2: フィルタリングルール例

通信方向	TCP	UDP	その他
内部から外部	許可	不許可	SMB は不許可
外部から内部	SMTP のみ許可	不許可	

以下，FreeBSD [11]，NetBSD [12]，Linux [13] に標準で付属するファイアウォールソフトによる，図 2.4のネットワーク構成におけるパケットフィルタリングルールを例示する．また汎用 OS を用いたファイアウォールシステムだけでなく，ルータ専用機によるファイアウォールによる構成も例示する．ルータ専用機としては Cisco Systems 社 [14] のルータを取り上げる．Cisco 社のルータでは，IOS(Internetwork Operating System) と呼ばれるソフトウェアによって，パケットフィルタリングの管理が行なわれる．本研究の対象とする大規模ネットワークにおいては，複数のパケットフィルタリングの実装が運用されており，実際に各実装でどのようにフィルタリングルールが表現されるかを述べる必要がある．

図中のフィルタリングルールに付加された番号は，同じ番号のルールが，それぞれの実装でほぼ同等のフィルタリングを行なうことを意味する．また，各フィルタリングルールの比較を行なうため，冗長に表現される部分がある．

各ファイアウォールの実装として以下のものを取り上げた．

- FreeBSD で動作する IPFW
- NetBSD で動作する IP Filter [15]
- Linux で動作する Netfilter [16]
- Cisco 社 IOS による独自のファイアウォール機構

これによって実際のパケットフィルタリングが，実装によってどのように表現されるかを述べる．

```
# 0
add pass ip from any to any via if1
# 1
add deny udp from any 137-139 to any
add deny tcp from any 137-139 to any
add deny udp from any to any 137-139
add deny tcp from any to any 137-139
# 2 % 両方向の SYN/ACK を通す
add pass tcp from any to any established
# 3
add pass tcp from any to any 25 setup
# 4
add deny log tcp from any to any in via if0 setup
# 5 % 内から外への SYN を通す
add pass tcp from any to any setup
# 6
add deny ip from any to any
```

図 2.5: FreeBSD 標準の IPFW

FreeBSD では，FreeBSD Project によって開発された IPFW を利用できる．設定は，コマンドラインからフィルタリングルールを追加するか，設定ファイルを読み込むことによって行なう．

```
# 0, 5
pass in quick on if1
pass out quick on if1
# 6
block in on if0 all head 100
# 1
block in quick from any port 136 >< 140 to any group 100
block in quick from any to any port 136 >< 140 group 100
# 2
pass in quick proto tcp all flags A/A group 100
# 3
pass in quick proto tcp from any to any port = 25 flags S/SA group 100
# 4
block in log quick proto tcp all flags S/SA group 100
# 6
block out on if0 all head 200
# 2
pass out proto tcp all flags A/A group 200
# 5
pass out proto tcp all flags S/SA group 200
```

図 2.6: NetBSD 標準の IP Filter

NetBSD は，外部で開発された IP Filter をシステム標準のファイアウォールとして採用した．IPFW とは違い，設定はフィルタリングルールをファイルから読み込むことで行なう．

```
# 0, 5
iptables -A INPUT -i if1 -j ACCEPT
# 1
iptables -A INPUT -p tcp --sport 137:139 -s 0/0 -d 0/0 -j DROP
iptables -A INPUT -p udp --sport 137:139 -s 0/0 -d 0/0 -j DROP
# 2
iptables -A INPUT -p tcp --tcpflags ACK ACK -j ACCEPT
# 3
iptables -A INPUT -i if0 -p tcp --syn --dport 25 -j ACCEPT
# 4
iptables -A INPUT -i if0 -p tcp --syn -j DROP
# 6
iptables -A INPUT -j LOG
iptables -A INPUT -j DROP
# 2
iptables -A OUTPUT -i eth0 -p tcp --tcpflags ACK ACK -j ACCEPT
# 5
iptables -A OUTPUT -i eth0 -p tcp --syn -s 0/0 -d 0/0 -j ACCEPT
# 6
iptables -A OUTPUT -j LOG
iptables -A OUTPUT -j DROP
```

図 2.7: Linux 標準の iptables

最新の Linux では iptables がシステム標準のファイアウォールとして利用できる．設定は，コマンドラインからフィルタリングルールを追加することで行なう．

```
interface if0
  ip access-group 100 in
  ip access-group 200 out
interface if1
!
# 1
access-list 100 deny   tcp any any range 137 139
access-list 100 deny   udp any any range 137 139
# 2
access-list 100 permit tcp any any established
# 3
access-list 100 permit tcp any any eq 25 syn
# 4
access-list 100 deny tcp any any syn
# 2
access-list 200 permit tcp any any established
# 5
access-list 200 permit tcp any any syn
# 6
access-list 200 deny ip any any log
```

図 2.8: Cisco IOS によるファイアウォール

Cisco 社のルータでは access-list という形でフィルタリングルールを設定し，そのフィルタリングルールをインターフェースに適用する．

このようにファイアウォールの実装間において，そのフィルタリングルールの表現は大きく異なる．複数のファイアウォールシステムを使用するような大規模なネットワークではこのことが，管理上の大きなコストとなる．詳しくは 3章で述べる．

2.2.2 IPsec におけるパケットフィルタリング

次に，能動的な安全性の確保を目的とするパケットフィルタリングである IPsec について述べる．

IPsec は，IP 層で暗号化・認証などの各機能を実現し，上位層を改変することなく通信の安全性を実現するための枠組みである．

本節では IPsec の概要を説明し，特にそのパケットフィルタリングシステムとしての側面について注目する．

IPsec の概要

IPsec は「IP 層で通信の安全性を実現する」という目的を達成するためのアーキテクチャであり，実際にはこの目的を実現するために複数のプロトコルが使われており，それぞれのプロトコルが協調動作することによって動く．

IPsec は主に以下の機能を実現する．

- 機密性
  受信者以外には，どんなデータを通信するか分からないように，データを暗号化する機能．ESP(Encapsulated Security Payload) [17] によって実現した．
- 認証 (正真性/確実性)
  通信相手の認証を行ない，相手が詐称をしていないことを保証する機能を提供する．AH(Authentication Header)[18] によって実現した．
- 完全性
  データが通信系路上で改ざんされないように，認証データを付加して通信を行なう．これは，ESP，AH によって実現した．
- 再送保護
  ある通信の記録をとっておき，その記録をそのまま再生することによって，記録をしたときに行なわれていた通信を再現しようとする攻撃をリプレイ攻撃と呼ぶ．たとえば，認証のための通信の記録をとっておき，悪意のある第三者がその通信をそのまま繰り返すことによって，認証を通過しようとすることなどが，それにあたる．IPsec の基本機能である再送保護では，パケットごとにシーケンス番号を付加することによってこの種の攻撃に対処する．

これらはすべて，IP 層で実現されており，telnet や POP といったアプリケーションは上記の安全性が保障されることを意識することはない．ちょうど，図 2.9のように，通信を行なう二点間に直接トンネルを張るようなものであり，アプリケーションのデータはこのトンネルを通ることで上記の保護を受けることができる．

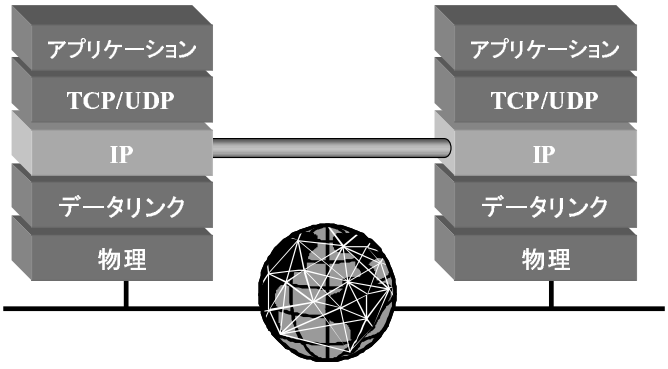

図 2.9: IPsec の概念図

IPsec もファイアウォールと同様，IPsec の機能を有するゲートウェイを通過するパケットの属性にもとづいて，当該パケットに対して IPsec による保護を適用するか否かの判断を行なう．

判断を行なう際に使われるパケットの属性には，以下のようなのもが挙げられる．

- 送信元アドレス
- 送信先アドレス
- 上位層プロトコル
- 利用しようとするサービス
- ICMP タイプ

IPsec ではこれらの属性にもとづいて，IPsec による保護が必要となった場合，セキュリティアソシエーション (SA) と呼ばれる仮想的なコネクションを確立する．SA の作成には管理者が静的に設定する手動鍵交換と自動鍵交換があるが，本研究では今後 IPsec を運用する上で一般的になる，自動鍵交換を対象とする．

自動鍵交換は，IPsec フィルタリングルールに該当するパケットがあった場合に，IPsec とは別に定義された自動鍵交換機構を利用して行なわれる．この際，IPsec を利用するか否かを決定するのは，フィルタリングルールである．

IPsec を実装したシステムでは通常，SAD(Security Association Database) と SPD(Security Policy Databe) という二つのデータベースをカーネル内 (もしくはそれに相当する部分) に持つ．SAD は生成された SA を保持するデータベースであり，ここに自動鍵交換で生成された SA が登録される．SA には以下の情報を保持する．

- 対象とするパケットの送信元/送信先アドレス
- セキュリティ・パラメータ・インデックス
- SA の生存期間
- 暗号鍵
- 暗号アルゴリズム

IPsec における実際のパケットの処理に必要な情報は SAD に保持されており，どのパケットを処理するかというフィルタリングルールは SPD に保持されている．

なお，自動鍵交換機構と IPsec におけるセキュリティアソシエーションの自動生成は本研究の対象外である．

IPsec のさまざまな実装

IPsec を利用したもファイアウォールと同じように，現在さまざまな実装がある．ここでは，IPsec におけるパケットフィルタリングのルールがどのように表現されるかを見る．

本節では図 2.10の構成のネットワークにおいて IPsec を運用するものとする．

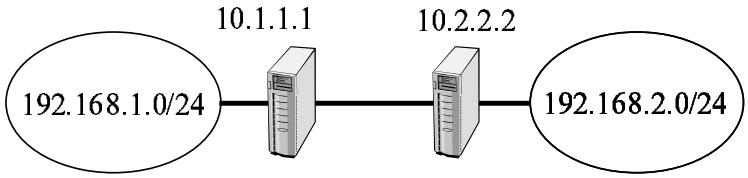

図 2.10: トポロジー

このネットワークにおいて，IPsec はトンネルモードの ESP を利用する．
初めに，図 2.11に KAME [19] における IPsec の設定例をしめす．

10.1.1.1 の設定

```
spdadd  192.168.1.0/24[any] 192.168.2.0/24[any] any
    -P out ipsec esp/tunnel/10.1.1.1-10.2.2.2/require ;
```

10.2.2.2 の設定

```
spdadd  192.168.2.0/24[any] 192.168.1.0/24[any] any
    -P out ipsec esp/tunnel/10.2.2.2-10.1.1.1/require ;
```

図 2.11: KAME におけるフィルタリングルールの表現

次に OpenBSD における IPsec の設定例を図 2.12に示す．

10.1.1.1 の設定

```
ipsecadm flow -proto esp -dst 10.2.2.2 -spi 101 -require
    -addr 192.168.1.0 255.255.255.0
          192.168.2.0 255.255.255.0
```

10.2.2.2 の設定

```
ipsecadm flow -proto esp -dst 10.1.1.1 -spi 102 -require
    -addr 192.168.2.0 255.255.255.0
          192.168.1.0 255.255.255.0
```

図 2.12: OpenBSD におけるフィルタリングルールの表現

Linux における IPsec の実装である FreeS/WAN [20] の設定例を図 2.13に示す．

10.1.1.1 の設定

```
conn ipsec
  type=tunnel
  esp=3des
  left=10.1.1.1
  leftsubnet=192.168.1.0/24
  leftnexthop=\%defaultroute
  right=10.2.2.2
  rightsubnet=192.168.2.0/24
  rightnexthop=\%defaultroute
  auto=start
```

10.2.2.2 の設定

```
conn  ipsec
  type=tunnel
  esp=3des
  left=10.2.2.2
  leftsubnet=192.168.2.0/24
  leftnexthop=\%defaultroute
  right=10.1.1.1
  rightsubnet=192.168.1.0/24
  rightnexthop=\%defaultroute
  auto=start
```

図 2.13: FreeS/WAN におけるフィルタリングルールの表現

最後に Cisco における設定を図 2.14に例示する.

10.1.1.1 の設定

```
interface if0
 ip address 10.1.1.1 255.255.255.0
 crypto map ipsec
!
interface if1
 ip address 192.168.1.1 255.255.255.0
!
access-list 100 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
!
crypto ipsec transform-set
 mode tunnel
crypto map ipsec 10 ipsec-isakmp
 set peer 10.2.2.2
 set transform-set esp
 match address 100
```

10.2.2.2 の設定

```
interface if0
 ip address 10.2.2.2 255.255.255.0
 crypto map ipsec
!
interface if1
 ip address 192.168.2.1 255.255.255.0
!
access-list 100 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
!
crypto ipsec transform-set
 mode tunnel
crypto map ipsec 10 ipsec-isakmp
 set peer 10.1.1.1
 set transform-set esp
 match address 100
```

図 2.14: Cisco におけるフィルタリングルールの表現

IPsec の実装においても，ファイアウォールにおける実装と同じように，同じ目的をもったフィルタリングルールでも，実装間において大きな差異が見られる．このことは複数のパケットフィルタリング機構を管理しなければならない管理者にとって大きな負担となる．

2.3　セキュリティポリシー

コンピュータネットワークにおける「セキュリティポリシー」はさまざまなレベルで解釈されることが多い.「あるネットワークにおける利用規定」であったり,「アクセスコントロールかけ方」であったりする．広義の意味でのセキュリティポリシーとは，自組織内のセキュリティを守るための行動規範であるといえる．

セキュリティ対策を完璧にすること，固定してしまうことは不可能である．なぜなら，日々のセキュリティ技術は進歩しており，また今まで安全と思われていた技術が，ある日脆弱性が発見され，使えなくなってしまうこともあるのである．また，逆に非常に厳しいセキュリティ対策をする (たとえば，外部との接続を絶ってしまう) ことによって，セキュリティを強固にすることも可能ではあるが，それでは利用者の利便性を損なってしまい，本来，利用者の利便性を向上させるはずの情報システムの目的と相反してしまう．従って，セキュリティ対策は継続的に行ない，今までのシステムの更新を行なっていかなければならないのである．

しかし，その場その場で発生するセキュリティインシデントに場当たり的に対応していたのでは，システム全体としての統一性を損なってしまうことになる．たとえば，ある領域では許可されていたことが，他の領域ではとくに正当な理由もなく許可されるという状態である．

そこで，管理者は自組織の情報システムの包括的な指針となる「セキュリティポリシー」を策定する必要がある．この場合のセキュリティポリシーとは，ネットワークにおけるアクセス制限という非常にシステムの実装に近い部分から，社内文書の取り扱いといった行動規範や，セキュリティシステムの監査といった，情報管理の方向性といったものまでを含んでいる．

本研究の対象とする，ネットワークにおけるセキュリティポリシーだけを見ても，アクセス制限，ユーザ管理・パスワード管理，不正アクセス監視，ログ管理など，その対象とするところは多岐に渡る．

しかし，この場合でも最も基本となるのはネットワークにおけるアクセス制限である．ユーザ管理・パスワード管理は本研究の対象から外れるが，その他のアクセス制限，不正アクセス監視，ログ管理などは，どれもアクセス制限にもとづく．

そこで本研究の対象とするネットワークにおけるセキュリティポリシーを，**「あるネットワークにおいて，いかにデータの流れに制限を適用するかを規定したもの」**と位置付ける．これは，ネットワークにおけるあるデータの流れが，その中間点においてゲートウェイによって監視されており，その監視内容によってルータゲートウェイの処理が変わるということを意味している．

ネットワークにおけるアクセス制限はもちろんのこと，たとえば不正アクセス監視なども，管理者の望まないアクセスである不正アクセスを監視するという意味で，広義のアクセス制限ととらえることができる．また，アクセス制限をかけた場合のログ管理は，適切なアクセス制限が適用されるかを検証するという点で，アクセス制限にとっても非常に重要なものである．さらに，アクセス制限を破られてしまった場合，どのような対策をとればよいかといった，セキュリティインシデント対策もアクセス制限の一環と言ってよいと考える．

2.4　セキュリティポリシーの運用

ではその上で，実際のネットワーク上でセキュリティポリシーがどのように実現されるかを検証する．

先に述べたように，本研究におけるネットワークセキュリティポリシーとは「あるネットワークにおいて，いかにデータの流れに制限を適用するかを規定したもの」である広義の意味でのアク

セス制限は，現在多くの場合，パケットフィルタリングのフィルタリングルールという形で実現される．

2.1.2節でも触れたように，現在ネットワーク上の多くのアクセス制限はファイアウォールで行なわれる．もちろん上に述べたようにセキュリティポリシーはアクセス制限だけで完結するものではないが，その基本的な部分を担う．

したがって，

$$\text{セキュリティポリシー} \supset \text{アクセス制限} \equiv \text{パケットフィルタリング}$$

とすることができる．では，パケットフィルタリングにおいて**実際**にどのようにセキュリティポリシーが実現されるかについて考察する．2.2.1節で述べたように，実装の段階ではアクセス制限は，対象とするパケットの定義と，対象となったパケットに対する処理という組み合わせで構成する．したがって，セキュリティポリシーとは，システムの実装の段階ではパケットフィルタリングのルールという形で表現されることが分かる．

では次に，2.1.1節で述べた能動的な安全性とセキュリティポリシーの関係について述べる．セキュリティポリシーの目的の一つに，情報のフローとその形式を定義するということがある．これは，どのように，どんな経路を通ってネットワーク上をデータが流れ，その際どのような形式でデータが配送されるかということを意味する．この場合の，データ形式は最も単純なかたちでは，平文であるか暗号文であるかといった形式が考えられる．

この場合もアクセス制限の場合と同じように，通常データフローの中間地点で暗号化等の処理が行なわれる．これは，2.1.1節で述べた，管理コストの問題のほかに，セキュリティーポリシーをどのように適用するかといった問題と関連する．

当然であるが，セキュリティポリシーは規定しただけではまったく意味を持たない．あるセキュリティポリシーを規定した上で，実際にそのセキュリティポリシーを情報システム上で有効にするポリシーエンフォースメントの存在が必要となる．

能動的な安全性を実現する上でのポリシーエンフォースメントには，

- エンドユーザにおけるポリシーエンフォースメント
- 自組織内の基幹網におけるポリシーエンフォースメント

という二通りの手法が考えられる．しかし，エンドユーザにおいてポリシーエンフォースメントを行なうためには，管理者が膨大な情報機器を管理する必要性が発生する．したがって，現在ではエンドユーザは意識することなく，ネットワークの基幹網においてポリシーエンフォースメントが行なわれるのが一般的である．

これは，基幹網におけるゲートウェイにおいてパケットフィルタリングを行ない，パケットの属性にしたがってポリシーエンフォースメントが行なわれていることを意味する．

受動的な安全性を実現するファイアウォールの例と同じように，能動的な安全性を実現する場においても，データフローの中間点でパケットフィルタリングが行なわれる．

能動的な安全性を実現するための一技術である IPsec も，このように動作しする．IPsec には，エンドツーエンドでの暗号通信を行なうトランスポートモードと呼ばれる通信形態もあるが，現在広く使われるのはトンネルモードと呼ばれる通信形態である．

トンネルモードでは，エンドノードから送られてきたパケットを，IPsec の動作するゲートウェイ (セキュリティゲートウェイ) でカプセル化することにより暗号化・認証を行ない，送信先のセ

キュリティゲートウェイでカプセルから再びもとのパケットを復元し，送信先のエンドノードに送信する．

この際，どのパケットをカプセル化するのか，どのようにカプセル化するのかといった属性がフィルタリングルールとして表現されており，これがセキュリティポリシーのポリシーエンフォースメントとして動作する．

安全な通信環境の確保には，その目的・通信の方向によって受動的な安全性と能動的な安全性があると述べたが，ここまで見たように，実際に安全性を実現する段階においては，どちらもパケットフィルタリングのルールという形式で表現される．さらに，ネットワークにおけるセキュリティポリシーの多くは，このパケットフィルタリングの形式で記述することができるのである．

したがって，本研究の対象とするネットワークにおけるセキュリティポリシーの管理とは，能動的な安全性を実現するパケットフィルタリング，受動的な安全性を実現するパケットフィルタリングの両システムのおけるフィルタリングルールの管理と等価であると定義する．

また，能動的な安全性を実現するパケットフィルタリング，受動的な安全性を実現するパケットフィルタリングとも，そのフィルタリングルールによってセキュリティポリシーを表現するという点において等価であると言える．

第3章　大規模ネットワークにおけるポリシー管理機構の問題

前章では，ネットワークにおける安全性を，その目的によって分類し，今日のネットワークセキュリティにおいてパケットフィルタリングが広く用いられていることを述べた．

また，それぞれの分野で広く用いられるパケットフィルタリング技術である，ファイアウォールと IPsec を説明し，実際に利用されているフィルタリングルールを例示し，本研究で対象とするセキュリティポリシーを，フィルタリングルールとして扱うことを述べた．

本章では，セキュリティポリシーとしてフィルタリングルールを運用していく上での問題点を示す．

3.1　セキュリティポリシーの管理

本節では，セキュリティポリシーがどのように管理されるかを述べる．

本研究が対象とする大規模ネットワークにおいては，セキュリティポリシーの管理は二つの側面を有する．それは次の二点である．

- 単一ノード (ゲートウェイ) におけるセキュリティポリシーの管理
- ネットワーク全体でのセキュリティポリシー管理

単一ノードにおけるセキュリティポリシーの管理としては，次のものが挙げられる．

- 新たなセキュリティリスクに対する対応
- 不要となったセキュリティポリシーの更新・破棄
- セキュリティポリシーのバージョン管理 (更新されたセキュリティポリシーに問題が発見されったときの，ロールバックの保障など)
- セキュリティポリシーの監査
- ログ管理

これらの管理はポリシーエンフォースメントとなる，ルータ・ゲートウェイが少数であれば問題ないが，大規模ネットワークにおいて管理対象が増えるに従って，個々のノード管理コストの総計は増加する．

さらに，管理対象が多数になるにしたがって，セキュリティポリシーをルータ・ゲートウェイにインストールするためのコストも増加する．管理対象が少数の場合は，管理者が手動でセキュリティポリシーを逐一インストールしても大きなコストにはならないが，多数になるにしたがい，これは大きなコストとなっていく．

次に，ネットワーク全体でのセキュリティポリシーの管理としては，次のものが挙げられる．

- 複数のセキュリティポリシーの整合性の確保
- 複数のログ情報の検証

これらの問題は複数のパケットフィルタリング機構が，同一の管理ネットワークで動作しなければならないときに顕在化する．

管理者は，あるゲートウェイで行なったフィルタリングルールの変更が，他のネットワークに存在するゲートウェイのフィルタリングルールと不整合を起こしていないかを確認しなければならない．これは，実際には他のフィルタリングルールを逐一確認し，行なった変更と付き合わせるという作業になる．

また，パケットフィルタリングを行なう場合，パケットフィルタリングが有効に機能するか，何か問題は発生していないかといったことを確認するためにも，ログの検証は必須である．しかし，管理対象が多数になると，この作業は非常にコストの高い作業となる．

3.2 セキュリティポリシー表現の不統一

セキュリティポリシー管理の煩雑さの多くの部分は，フィルタリングルールの設計と実装，そして実装されたフィルタリングルールの定期的な検査を行なわなければならないことによる．図3.1はセキュリティポリシー管理のサイクルを表す．

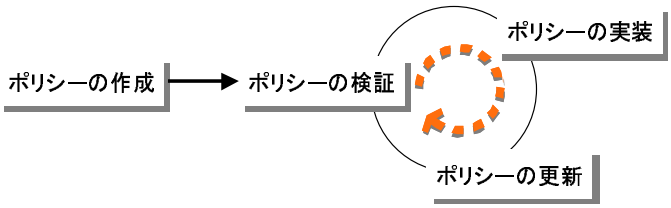

図 3.1: 管理サイクル

ポリシー作成の段階では，自組織内のネットワークの設計に合わせて，どのようなアクセス制限を行なうかを決定する．

次にネットワーク全体のセキュリティポリシーを集約し，セキュリティポリシーの検証を行なう．検証段階では，作成したセキュリティポリシーが自組織内の要求事項を満たすか，それぞれのセキュリティポリシーが衝突をおこしていないかなどを検証する．

最後に，作成されたセキュリティポリシーをフィルタリングルールのかたちで実装を行なう．

実装が行なわれ，運用に入ると，セキュリティポリシーに対する要求の変更や，新たなセキュリティリスクの発生，セキュリティ技術の開発などにより，初めに作成したセキュリティポリシーの更新が必要となる．セキュリティポリシーを更新した場合，もう一度検証から行ない，実装へと進んでいく．

本研究はパケットフィルタリング機構を実際にネットワーク上で管理・運用する際の問題を対象とするため，セキュリティポリシーの作成，セキュリティポリシーの検証の段階では，本研究の対象とする管理の問題は発生しない．

次の実装の段階において，以下の問題が発生する．

ここで，まず問題となるのは，セキュリティポリシーの表現方法であるパケットフィルタリングルールの記述方法の各パケットフィルタリング機構間での差異である．2.2節で述べたように同一目的をもったパケットフィルタリング機構間，つまり同じファイアウォールの間においてでさえ，異なった記述方法を利用している．

また，大規模ネットワークにおいてパケットフィルタリングを行なう場合，IPsec とファイアウォールのように，異なる目的をもったシステム間でのパケットフィルタリング行なうことも多いが，同じようにフィルタリングルールの記述方法は異なる．

しかし，パケットフィルタリング機構の利用者は，フィルタリングルールの習得が目的ではなく，セキュリティポリシーを実現することが目的である．したがって，あるセキュリティポリシーをどのようにフィルタリングルールで表現するかということを，それぞれの実装ごとに考えなければならないのは，利用者にとって大きな負担となる．

これによって，大規模ネットワークでのパケットフィルタリング機構の普及が阻害されてしまい，基本的な安全性を達成できない．

またこのことによって，単一の実装がネットワーク全体で利用される傾向にある．単一の実装を使うことによってフィルタリングルールの不統一の問題は解決されるが，依然，多数のフィルタリング機構を運用する上での問題は残る．さらに，選択肢が限定されることによって，実装の制限に縛られ，目的に応じたシステムを組むことができない．

3.3 セキュリティポリシーの配布における問題

次に分散環境でセキュリティポリシー管理を行なう上での問題について述べる．

ここで，ファイアウォールを例にとる．ファイアウォールはファイアウォール専用機が多く用いられており，ユーザインターフェースは，その専用機独自である．

2.2.1節で挙げた Cisco 社のファイアウォールは，すべて CLI (Command Line Interface) で設定を行なう．設定を行なうゲートウェイに接続し，2.2.1節で例示した設定をコマンドラインからすべて入力する．広く分散したファイアウォールを一台ずつ設定しなければならず，ネットワーク全体のセキュリティポリシーを変更する場合，実装に非常に時間がかり，さらに誤ったセキュリティポリシーを設定する可能性が高くなる．

また，ファイアウォール・ソフトウェアの分野で 41%のシェアを占める [21]CheckPoint 社の製品である Firewall-1 は GUI (Graphical User Interface) で設定を行なうが，この場合でも，それぞれのファイアウォールに接続し，一つ一つ設定を行なわなければならない．Firewall-1 の操作画面の例を図 3.2に例示する．

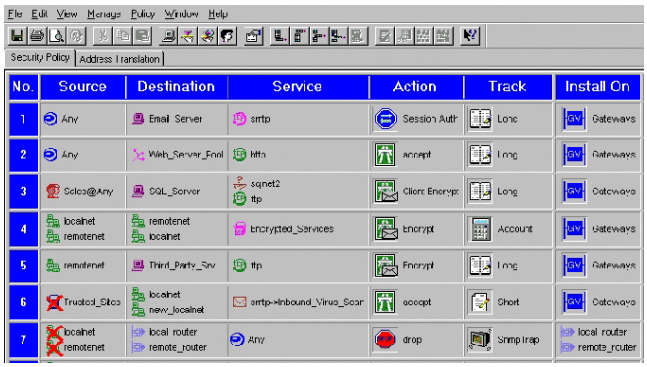

図 3.2: FW-1 の設定画面 (引用 [1])

CLI に比べてユーザに分かりやすくなってはいるが，たとえばフィルタリング対象ホストを追加する操作は，図 3.3に示したダイアログから，一台一台追加しなければならず，やはり大規模なネットワークでセキュリティポリシーの変更を行なう場合のコストは高い．

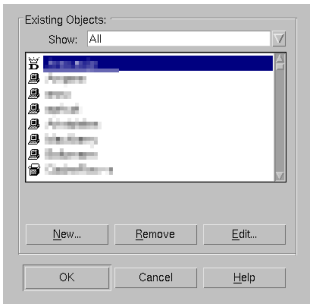

図 3.3: 対象ホストの追加

次に，同じ 2.2.1節で取り上げた PC-UNIX で実装されたファイアウォール製品を見ると，Cisco 社の製品と同じように CLI が主体となっており，ファイアウォール一台一台に接続し，フィルタリングルールを書き換えた上で，フィルタリングルールのインストールをする，という作業を行なわなければならない．

IPsec におけるポリシー管理もファイアウォールでの状況と同じく，自動鍵交換による SA の動的な生成は可能だが，セキュリティポリシー配布のための一般的なプロトコルが規定されていないため，セキュリティポリシーに基づいて IPsec を利用することが不可能である．

これは，たとえすべてのパケットフィルタリング機構を同一製品で構成したとしても解決されない問題である．したがって，現在，大規模ネットワークにおいて，広域に分散したパケットフィルタリング機構を一元的に管理するのは，非常に困難である．

3.4　セキュリティポリシー間の不整合

最後の問題点として，各パケットフィルタリング機構の間での，セキュリティポリシーの整合性の確保の問題が挙げられる．

例として，図 3.4のような場合を考える．

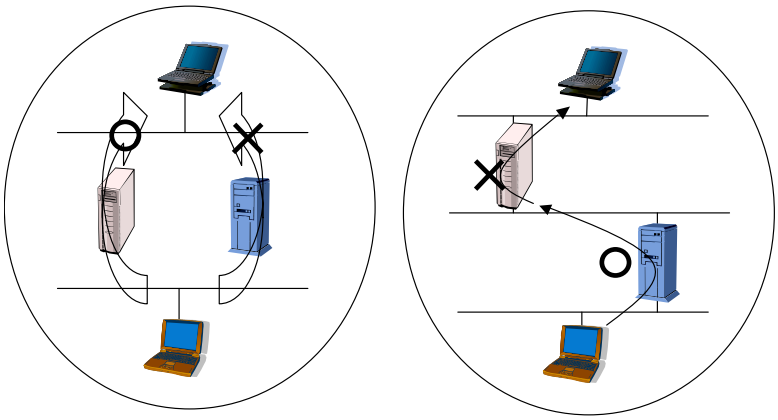

図 3.4: セキュリティポリシーの衝突

図の左の例では，片方は送信先ホストへの通信を許可しており，もう片方が禁止している．送信先ホストへのゲートウェイとなるルータのセキュリティポリシーが一環していないため，経路制御の結果によって送信先ホストへ到達できる場合とできない場合が発生してしまう．

また，図の右の例では，上流のゲートウェイは送信先ホストへの通信を禁止しているが，下流のゲートウェイは送信先ホストへの通信を許可している．このようなセキュリティポリシーは現実的にはあると思われるが，この場合下流のゲートウェイが通信を許可しているのは，冗長な設定であると言える．

また，下流からの通信を考慮に入れず，上流でこのようなアクセス制限をかける場合も考えられる．

このように，パケットフィルタリングは，あるデータフローの中間点で行なわれており，始点から終点までで一貫したポリシーを提供していなければ，本来可能であるはずの通信が禁止されたり，逆に，許可されるべきでない通信が可能になってしまう可能性もある．

このセキュリティポリシーの整合性の問題は，パケットフィルタリングが利用されるネットワークが，大規模になればなるほど深刻になる．

さらに，ファイアウォール間のセキュリティポリシーの整合性だけでなく，同じパケットフィルタリング機構である IPsec のセキュリティポリシーとの整合性も保障する必要がある．

この問題によって，セキュリティポリシーの変更が予期せぬ通信断をもたらすことや，セキュリティホールの発生につながることがある．

3.5　まとめ

本章を簡単にまとめると以下の通りである．

- パケットフィルタリングは実際にいくつかの目的でおこなわれており，その目的によって，システム構成も異なる．しかし実際に人間が設定する段階では，そのパケットフィルタリングの目的によらず，ほぼ同じような設定をする．したがって，似たような設定を複数繰り返すことは，パケットフィルタリングを行なうネットワークが大規模になるほど，運用コストを高める．

- 大規模ネットワークにおけるパケットフィルタリングの運用の上で，コストの主な要因となるものに，各パケットフィルタリングの実装間で，フィルタリングルールの統一性がないことが挙げられる．そのため，複数の実装を運用する場合，管理者は各フィルタリングルールに精通する必要がある．

- ある同一ネットワーク内で，セキュリティポリシーは統一していなければならない．セキュリティポリシーの統一のためには，パケットフィルタリングのルールの統一性が不可欠であるが，その整合性を保つのは困難である．

- ある同一ネットワーク内で，統一的なセキュリティポリシーを実現する上で問題となるのが，セキュリティポリシーを配布するための汎用的な仕組みがないことである．そのため，各実装間で，セキュリティポリシーを統合することができない．

これらの問題は，既存のパケットフィルタリング機構自身にあるわけではなく，複数のパケットフィルタリング機構を単一のネットワークにおいて運用するときに顕在化する．

現在，ネットワークセキュリティ分野の進展により，パケットフィルタリングは広く利用されるようになっており，この問題が大きなものとなる．

第4章　統一的なポリシー管理機構

前章では現在，ネットワークのさまざまな地点において行なわれるパケットフィルタリングを運用する上での問題点についてまとめた．

本章では，3章で挙げた問題を解決するために，以下の二つの手法を用いたシステムを提案する．

- セキュリティポリシー表現の統一的フォーマットの作成
- セキュリティポリシーデータの蓄積配布機構の構築

さらに，実際のネットワークで本システムを運用するために必要な機能について説明する．

- 管理情報の収集
- 各機構の独立性の保障

上に挙げた二つの手法についてそれぞれ，目的と実現できる機能，そしてそれによって解決できる問題について述べる．さらに次に挙げた二つの機能の必要性とその目的について述べる．

4.1　セキュリティポリシー表現の統一の必要性

本研究では，汎用的なフィルタリングルールを定義することを提案する．汎用的なフィルタリングルールを定義することによって，パケットフィルタリングの利用者は，本システムを使うことによって，一度フィルタリングルールの記述方法を習得すれば，異なるパケットフィルタリング機構を使う場合でも，同一のフィルタリングルールを使うことができるようになる．その結果，複数のパケットフィルタリング機構を利用する場合でも，統一的な設定を行なうことができる．

そこで本節では，以下の流れに沿って，一般的なフィルタリングルールの定義を行なうための手法について述べる．

- まず初めに，異なる実装間におけるセキュリティポリシー表現の共通部分について述べる．
- パケットフィルタリングにおいて必要となる属性情報に関する考察を行なう．

4.1.1　セキュリティポリシー表現の共通性

2.2節で見たように，現在のセキュリティポリシーの表現であるフィルタリングルールには各実装によって差異が大きい．しかし，共通する部分も少なからず見られる．実装によって「データの流れに制限をかける」フィルタリングそのものの実現方法はさまざまであるが,「どのデータにフィルタリングをかけるか？」といったポリシーに対応する部分は共通の部分が多い．

そこで，セキュリティポリシーの汎用的表現の定義を行なう前に，まず各実装における共通部分の検討を行なう．これを行なうことによって，既存のフィルタリングルールの利用者が，必要以上の学習コストをかけることなく，本システムを利用することができるようになる．

ここでは，2.2節で言及したフィルタリング機構の中からいくつかを選択し，そのフィルタリングルールを検討する．選択の基準は，

- 多くのプラットフォームで利用できること
- 汎用 OS によるものだけでなく，専用機を含めること

とし，IP Filter，Cisco IOS，KAME を選択した．Cisco IOS に関しては，ファイアウォールと IPsec の二つのフィルタリングルールを検討する．

IP Filter

```
pass in quick on if1
pass out quick on if1
block in quick from any port 136 >< 140 to any group 100
block in quick from any to any port 136 >< 140 group 100
```

Cisco IOS (ファイアウォール)

```
interface if0
  ip access-group 100 in
  ip access-group 200 out
interface if1
access-list 100 deny   tcp any any range 137 139
access-list 100 deny   udp any any range 137 139
```

KAME

```
spdadd  192.168.1.0/24[any] 192.168.2.0/24[any] any
  -P out ipsec esp/tunnel/10.1.1.1-10.2.2.2/require ;
```

Cisco IOS (IPsec)

```
interface if0
  ip address 10.1.1.1 255.255.255.0
  crypto map ipsec
interface if1
  ip address 192.168.1.1 255.255.255.0
access-list 100 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
crypto ipsec transform-set
  mode tunnel
crypto map ipsec 10 ipsec-isakmp
  set peer 10.2.2.2
  set transform-set esp
  match address 100
```

図 4.1: フィルタリングルールの比較

まず分かることは IOS による IPsec の設定を除いて，他の三つはトラフィックの方向の概念を持つ．また，IOS による IPsec の設定も access-list の内容によって暗黙的に方向を持つ．

次に，フィルタリングルールの中心となる，パケットの選別を行なう部分の検討する．

IOS では明示的に access-list というキーワードによってフィルタルールが書かれる．その内容は，

- フィルタ番号．これは同じ番号のフィルタを一グループとして扱うために使用する．
- 許可/不許可
- プロトコル

- 送信元 IP アドレス (IP アドレスの範囲)
- 送信元ポート番号 (ポート番号の範囲)
- 送信先 IP アドレス (IP アドレスの範囲)
- 送信先ポート番号 (ポート番号の範囲)

である．これは，他の実装においてもほぼ同じ形式である．

ただし，範囲指定がそれぞれ少しずつ異なっており，ネットワークアドレスを表現するのに KAME, IP Filter ではネットワークマスク長で指定できるが，IOS ではサブネットマスクをビット反転したものを使用する．また，許可/不許可のキーワードも IOS と IP Filter で異なる．

4.1.2 セキュリティポリシーの汎用的表現

前節で，細かい部分では差異があるものの，フィルタリングルールの基本的な書式は近似していることをのべた．

したがって，本研究では特定のフィルタリングルールを選択し，次の条件を満たすための拡張を行なう．

- ファイアウォール，IPsec を同列に扱う
- 他の実装で使われているが，選択した書式にない記述の追加

この条件を満たすために，パケットの処理内容に，許可/不許可/IPsec を指定できる必要がある．また，IPsec の設定はフィルタリングルールとは別に設定する必要がある．

処理の流れは以下の通り．

1. フィルタリングルールによってパケットをマッチさせる
2. パケット処理の指定によって許可/破棄/IPsec の，いずれかの処理を行なう

ここで IPsec を指定した場合，IPsec 特有の属性 (セキュリティプロトコルの指定，モードなど) があるため，IPsec の設定を別にしなければならない．

4.1.3 既存システムへの適応機構の必要性

本研究では汎用的なセキュリティポリシー記述言語を定義するが，今日使われるさまざまな実装は当然，独自の記述方式を使う．したがって本機構を利用するためには，各実装における記述言語に変換する必要がある．

変換はセキュリティポリシーのインストールの際に行い，テキストを置換するフィルターで実現される．同時に，変換対象とするフィルタリングルールで使用できないルールのチェックも行なう．

実装によっては，ルールの並びなどの書式が大きく異なるので，変換は個別に行なう．

4.2　セキュリティポリシーデータの蓄積と配布

3章で述べたように現在ではさまざまな目的に応じて，複数の実装によってパケットフィルタリングが行なわれる．大規模ネットワークの実際の現場においては，通常，単一の実装ではなく，複数のシステムが協調し動作する．

この場合の協調は，同一目的をもったパケットフィルタリングシステム間の協調だけではなく，異なる目的をもったシステム間の協調もふくまれる．図 4.2に示すように，たとえば，複数のファイアウォール間の協調だけでなく，IPsec とファイアウォールといった目的を異にするシステム間の協調も考慮しなければならない．

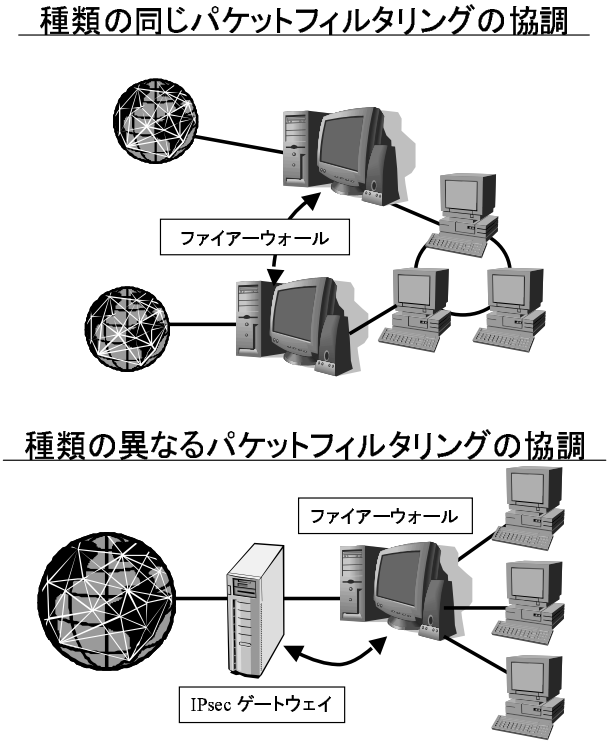

図 4.2: システム間の協調

このため，パケットフィルタリング機構の実装によらず，フィルタリングルールを一箇所に蓄積し，さらに各フィルタリングノードに配布するための機構が必要となる．

この問題の解決のために二つの手法を提案する．

4.2.1　セキュリティポリシーの一元的管理

セキュリティポリシーを一元的に管理するために，各フィルタリングノードのフィルタリングルールを蓄積するセキュリティポリシーリポジトリを定義する．

リポジトリはある管理単位のネットワークごとに設置するものとする．したがって他組織のネットワークとの接続ポイントで連携が必要となる場合は，本研究の対象外とする．

リポジトリには，4.1節で述べた記述言語によって記述されたセキュリティポリシーを保持する．

セキュリティポリシーは，フィルタリングノードごとに保持する．これは，単一のフィルタリングノードで，異なる種類のパケットフィルタリング (たとえばファイアウォールと IPsec) を行なう場合でも，フィルタリングポリシーを記述したデータは一つであることを意味する．

これによって管理者は分散環境でのセキュリティポリシーを，セキュリティリポジトリのみにおいて一元的に管理できる．

4.2.2 汎用的プロトコルを利用したセキュリティポリシー配布機構

セキュリティポリシーを集約したリポジトリから，実際にパケットフィルタリングを行なうノードに対してフィルタリングルールをインストールする必要がある．

この際，以下のことを前提とする．

- トランザクションベースとする
- 盗聴・詐称からの保護を保障する

通常，セキュリティポリシーリポジトリとフィルタリングノードの間の通信は，フィルタリングルールのインストール，フィルタリングノードからリポジトリへ通知を送る場合がほとんどである．通常二者間の通信はほとんど行われないため，常にコネクションを保持することは無駄である．また，常時コネクションがあることを仮定すると，ネットワークの障害などによってコネクションが切れた場合の処理などによって全体が複雑になってしまう．

次に，リポジトリとフィルタリングノード間の通信は，フィルタリングルールという，機密性の非常に高い情報をやりとりするため，盗聴・詐称によって攻撃を受けないようにする必要がある．

また，フィルタリングノードの実装上の制限により，利用できるプロトコルが制限される場合がある．たとえば，Cisco IOS を利用する場合，新たにプロトコルを追加することは，ほぼ不可能である．

そのような場合，本機構の利用するプロトコルを用いてリポジトリと通信を行ない，その結果をフィルタリングノードにインストールするための，代行者 (エージェント) が必要となる．エージェントとフィルタリングノードの間の通信は，別途通信の安全を確保する必要がある．通常は，シリアルケーブルで接続する，外部との接続性を持たないローカルネットワーク経由で接続する，特別に容易した IPsec のコネクションを利用するなどの方法によって確保できる．

4.3 管理情報の収集

管理情報とは主に，各フィルタリングノードによって収集されるログを意味する．

ネットワークセキュリティを管理する上では，セキュリティポリシーは設定するだけでなく，継続的にセキュリティポリシーの妥当性を検証する必要がある．

管理情報の蓄積にはセキュリティポリシーリポジトリを使う．これは収集したログを利用してセキュリティポリシーを検討することを容易にするためである．

これによって，管理者は複数のフィルタリングノードにアクセスすることなくリポジトリのみにおいてログの検証を行なえる．

また，各フィルタリングノードからリポジトリへのログの送信には，セキュリティポリシーの配布に使われるプロトコルと同じものを利用する．

4.4 各機構の独立性の保障

ある特定のフィルタリングノード (もしくはそのエージェント) が乗っ取られた場合，その被害が他のフィルタリングノードに波及しないようにする必要がある．

そのため，フィルタリングノードとリポジトリの間の通信は，リポジトリによってフィルタリングノードの認証を行ない，厳密なアクセスコントロールを行なう必要がある．

また，リポジトリが乗っ取られることも考慮する必要がある．そのため，できるだけ保持するデータも暗号化しておくことが望ましい．あるフィルタリングノードのデータは，そのフィルタリングノードの公開鍵と管理者の公開鍵で暗号化しておくことにより，第三者に解読される可能性は非常に低くなる．

ただし，バックアップの問題や，キーリカバリの問題もあるので，必ずしも暗号化する必要はない．リポジトリの十分な要塞化をすることが重要である．

第5章　ポリシーの管理と配布の実現

3章では，既存のセキュリティポリシー管理機構の問題を指摘した．4章ではこれらの問題を解決するための手法を提案した．

本章では，4章で提案した手法を用いて，統一的なセキュリティポリシー管理機構を実現する．

5.1　全体の構成

本節では，本機構の全体の構成と設計方針について説明する．

本機構は，全体として四つの構成要素から成る．

- 汎用的なセキュリティポリシー記述言語
- セキュリティポリシーリポジトリ
- セキュリティポリシーサーバとエージェント
- ログ管理機構

図 5.1に，これら構成要素の関係図を示す．

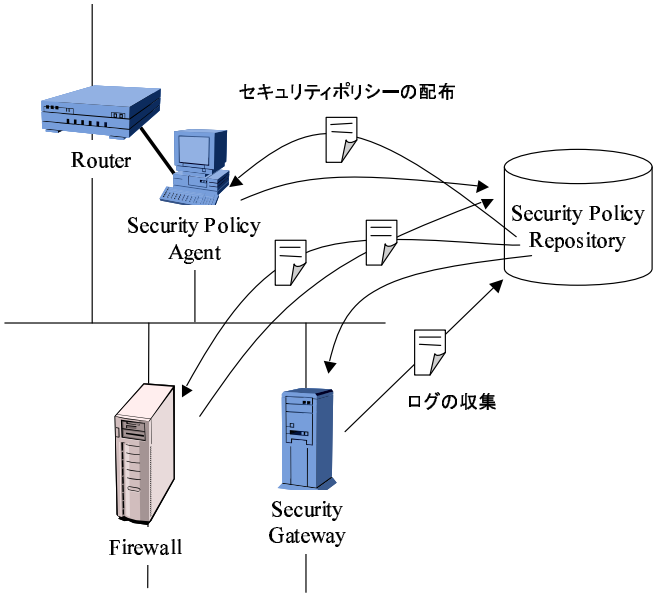

図 5.1: 全体構成

汎用的なセキュリティポリシー記述言語では，実装によって異なるフィルタリングルールを一般化し，複数の実装で統一された記述方式が使えるようにした．本研究ではまず基本的な命令を定義した．

セキュリティポリシーリポジトリは，汎用的なセキュリティポリシー記述言語で記述されたフィルタリングルールを蓄積する．また，同時にフィルタリングノードからのログなどの管理情報も蓄積する．管理者は，リポジトリに蓄積されたフィルタリングルールを変更し，変更結果をフィルタリングノードに送信し，セキュリティポリシーの変更を行なう．

セキュリティポリシーやログ情報の送受信を行なうのが，セキュリティポリシーサーバとセキュリティポリシーエージェントである．

通常，セキュリティポリシーサーバはリポジトリで動作しており，要求されたセキュリティポリシーの送信，ログ情報の受信などを行なう．

セキュリティポリシーエージェントは，フィルタリングノードで動作し，フィルタリングルールの受信，フィルタリングノードの実装に基づくルールの変換動作，ログ情報の送信などを行なう．ただし，フィルタリングノードが汎用 OS ではない場合，フィルタリングノードとは別のコンピュータで動作し，処理結果だけをフィルタリングノードに送信する．

ログ管理機構は，エージェントとリポジトリによって構成されており，フィルタリングノードが生成するログ情報を定型に変換し，リポジトリに保存する．

5.2 汎用的なポリシー記述言語

本節では，4.1節で述べた方針に従って定義した，ポリシー記述言語について述べる．

本機構のポリシー記述言語の基として，IPFW の記述方式を選択した．IPFW を選択した理由として以下を挙げる．

本機構におけるフィルタリングポリシーの書式は図 5.2の通りである．

```
action [log [logamount number]] proto from src to dst \
    [interface-spec] [options]
```

図 5.2: 汎用的なポリシー記述言語 (実際には一行)

- proto from src to dst [interface-spec] [options]
  「パケットフィルタリングに使用する属性の定義」を指定する

- action [log [logamount number]]
  「パケットフィルタリング後の動作」を指定する．

以下,「パケットフィルタリングに使用する属性の定義」と「パケットフィルタリング後の動作の定義」について述べる．最後に「IPsec の処理内容の定義」について述べる．

なお，下線部は必須の値，{a|b}は a か b かを必ず選択，カッコ内は省略可能，をそれぞれ示している．

5.2.1 パケットフィルタリングに使用する属性の定義

図 5.3に，フィルタリングに使用する属性を例示する．

proto from src to dst
`{in|out}` [interface-spec] [options]

図 5.3: フィルタルール

次に，各属性を説明する．

- proto
  パケットのプロトコル番号を指定する．'all' を指定するとすべてのプロトコルに適合する．
- src，dst
  パケットの送信元，送信先を示す．'IP アドレス'，'IP アドレス/マスク長'，'IP アドレス:ネットマスク' のいずれかで指定する．プロトコルとして TCP か UDP を指定した場合，`{port|port-port|,port[,...]}`のいずれかの書式でポート番号を指定できる．
- `{in|out}`
  'in' か'out' のどちらかを選択し，パケットが入力か出力かを示す
- `[interface]`
  パケットが通過するインターフェースを指定する．実装によっては必ずインターフェースの指定が必要である．
- `[option]`
  細かいオプションを指定する

5.2.2 パケットフィルタリング後の動作の定義

フィルタリング後の動作としては，次のものを定義する．

- allow
  何も処理せずそのまま転送
- deny
  パケットを破棄
- ipsec rule-num
  IPsec の処理をする．rule-num には IPsec の処理内容を記述した rule-num を書く
- log
  パケットの記録をとる
- reject
  パケットを破棄し，ICMP の host unreachable を送信

- reset
  パケットを破棄し、TCP の reset を送信．TCP のみを対象

オプションとして，[log [logamount number]] を指定した場合，回数 [logamount number] に達するまで，そのアクションにマッチしたパケットの記録をとる．

5.2.3 IPsec の処理内容の定義

図 5.4に IPsec の処理内容の定義を示す．

```
ipsec rule-num proto sec-proto level req-level [spi]
    peer src peer-src dst peer-dst
```

図 5.4: IPsec の処理 (実際には一行)

次に，各属性を説明する．

- rule-num
  action ipsec の時に指定するルール番号を指定する
- sec-proto
  IPsec のセキュリティプロトコルを指定する．'ah'，'esp'，'esp-auth' を指定できる
- req-level
  IPsec のレベルを指定する．'require'，'use'，'default' を指定できる
- peer-src, peer-dst
  IPsec トンネルモードのトンネルの両端のアドレスを指定する
- `[spi]`
  SPD に SPI を指定する必要のある実装の場合に SPI を指定する

5.3 セキュリティポリシーリポジトリ

本節ではフィルタリングルールを蓄積する，セキュリティポリシーレポジトリについて述べる．セキュリティリポジトリで管理すべき情報は次の三つである．

- 対象ノードの IP アドレス
- セキュリティポリシーの本文
- ログ情報

初めに，データベース内のデータ形式を定義し，データベースへの入出力について述べる．

5.3.1 データ形式

セキュリティポリシーは RDBMS を用いて管理する．すべてのフィルタリングノードのセキュリティポリシーを単一のテーブル ”secpolicy” で管理し，図 5.5のフィールドを持つものとする．

```
CREATE TABLE secpolicy (
    address INET,
    policy TEXT,
    log TEXT
);
```

図 5.5: テーブル

ここで，TEXT 型は可変長のテキストデータである．INET 型はネットワークアドレス情報を含む IP アドレスであり，表 5.1で示した表記で記述する．ネットマスク長を付加することで，同一ネットワーク内のフィルタリングノードを簡単に検索することができる．

表 5.1: INET 型

ホスト部	ネットマスク長
10.3.4.5	/24

すべてのセキュリティポリシーを単一テーブルで管理する点に関しては，一つの管理ネットワークに数百台以上ものフィルタリングノードはない．もしある場合，管理ネットワークを分割し，異なるポリシーで運用すべきであると考え，問題ではないとした．

5.3.2 操作インターフェース

データベースの入出力インターフェースは，通常のデータベース問い合わせと同じである．ユーザインタフェースとしては WWW を利用し，WWW ブラウザからクエリーを行なうことで，セキュリティポリシーの検索・閲覧・更新を行なう．図 5.6にいくつかの操作を例示する．

セキュリティポリシーの追加

```
INSERT INTO secpolicy VALUES (
  '10.3.4.5',
  'allow ip from any [...]',
  ''
);
```

セキュリティポリシーの削除 (ノードの削除)

```
DELETE FROM secpolicy
  WHERE address = '10.3.4.5';
```

セキュリティポリシーの更新

```
UPDATE secpolicy SET policy = '[...]'
  WHERE address = '10.3.4.5';
```

特定のノードのセキュリティポリシーの検索

```
SELECT address,policy FROM secpolicy
  WHERE address = '10.3.4.5';
```

同一ネットワークにあるセキュリティポリシーの検索

```
SELECT address,policy FROM secpolicy
  WHERE network('address') = network('10.3.4.5/24')
```

図 5.6: セキュリティポリシーの操作

これによって，ネットワーク全体のセキュリティポリシーを一覧することが可能となり，ネットワーク全体でのセキュリティポリシーの整合性を向上することができる．

5.4 ポリシーサーバとエージェント

本節では，セキュリティポリシーの送受信を行なう，セキュリティポリシーサーバとセキュリティポリシーエージェントについて述べる．また，二者の間で用いられるプロトコルについて述べる．

5.4.1 サーバ-エージェント間プロトコル

サーバ-エージェント間プロトコルとして HTTP を用いる．HTTP は SSL を使うことにより通信の暗号化，通信相手の認証などを容易に行なうことができ，またコネクション型のプロトコルと

して，ファイルの転送などに多く用いられているため採用した．

サーバ-エージェント間の通信はすべて SSL によって暗号化する．

サーバからクライアントへの通信，クライアントからサーバへの通信の双方向において SSL で使われる電子証明書による認証を行なう．ポリシーサーバや各エージェントはそれぞれ固有の電子証明書を持っており，電子証明書によってアクセス制限を行なう．これにより悪意のある者が，ポリシーサーバであることを偽って，虚偽のセキュリティポリシーを配布できなくなる．逆に，ポリシーサーバから，アクセス権のない者がセキュリティポリシーを取り出すこともできなくなる．

セキュリティポリシーの配布は，HTTP 標準のプロトコルである，PUT/GET を利用して行なう．

図 5.7は，ポリシーサーバからエージェントへ，セキュリティポリシーを配布する際の，トランザクションを示している．

サーバからクライアントへの送信

```
PUT /etc/secpolicy.conf HTTP/1.1
Host: 10.1.1.1:80
Authorization: Basic XXehng849sj9gjg=
Content-Length: 302
# start of filteing rule
allow ip from any to any
[...]
# end of filtering rule
```

クライアントからの確認 (成功時)

```
HTTP/1.1 201 Created
Date: Thu, 13 Dec 2001 10:14:23 JST
Location: http://localhost/etc/secpolicy.conf
Server: Apache/1.3.19 (Unix)
Content-Type: text/plain
<HTML><BODY>Transaction Completed</BODY></HTML>
```

クライアントからの確認 (失敗時)

```
HTTP/1.1 500 Internal Server Error
Date: Thu, 13 Dec 2001 10:14:23 JST
Location: http://localhost/etc/secpolicy.conf
Server: Apache/1.3.19 (Unix)
Content-Type: text/plain
<HTML><BODY>Transaction Failed</BODY></HTML>
```

図 5.7: サーバ・エージェント間の通信

サーバからクライアントへ，HTTP でアクセスし，PUT メソッドによりセキュリティポリシー

を送信する．PUT メソッドの引数は，フィルタリングノード内でセキュリティポリシーを保存する場所を示し，セキュリティポリシーの実体は標準入力として送信する．

セキュリティポリシーの配信を受けたポリシーエージェントは，PUT メソッドの結果に応じて応答を返す．エラーを受けた場合，ポリシーサーバはその旨を管理者に通知する．

5.4.2 ポリシーサーバ

ポリシーサーバは次のことを行なう．

- フィルタリングノードに対するセキュリティポリシーの配布
- 管理者からのセキュリティポリシーに対する操作の受付
- ポリシーエージェントからセキュリティポリシーを取得する
- ポリシーエージェントからのログ・アラートの受付

ポリシーサーバの主な目的は，管理者が設定したセキュリティポリシーをフィルタリングノードに配信することである．ポリシーサーバは HTTP サーバとして動作し，管理者は WWW ブラウザを通してセキュリティポリシーの操作を行なう．

また，ポリシーエージェントから受信したログを，フィルタリングノード毎に保持し，管理者の要求に応じて提示する．

5.4.3 ポリシーエージェント

ポリシーエージェントは次のことを行なう．

- ポリシーサーバからセキュリティポリシーの配布を受け付ける
- セキュリティポリシーの書式変換
- フィルタリングノードへのセキュリティポリシーのインストール
- ローカルのセキュリティポリシーをポリシーサーバに送信する
- ログの送信
- アラートの送信

ポリシーエージェントはセキュリティポリシーをポリシーサーバから受信する場合，HTTP サーバとして動作する．また，ポリシーサーバにアラートを送る場合などは，HTTP クライアントとして動作する．

ポリシーサーバからセキュリティポリシーを受信したポリシーエージェントは，フィルタリングノードの理解できる書式へとセキュリティポリシーを変換し，インストールする．このため，ポリシーエージェントは利用するフィルタリング機構の種類・実装に応じて用意しなければならない．

また，ポリシーエージェントはポリシーサーバへのログの送信を行なう．既存のフィルタリング機構はさまざまなログを生成する．ポリシーエージェントは生成されたログを**一定の間隔**でポリ

シーサーバに送信する．ログの生成間隔は非常に短い場合があるので，一定量のログが蓄積されてから送信すると，ログの送信コストを下げられる．

ログの追加は，HTTP 上で SQL を用いて行なう．図 5.8は，ログ追加操作の例である．'address' として自分の IP アドレスを指定し，ログを追加する．

ログの追加

```
UPDATE secpolicy SET log='[....]'
  WHERE address = '10.3.4.5';
```

図 5.8: ログの追加

5.5 本機構の評価

本節では，これまで述べてきた統一的なセキュリティポリシー管理機構について，既存のフィルタリングシステムとの定性的評価について述べる．

まず，本機構の比較対照となる機構を示す．その上で，どのような項目について本機構と比較対照を比較し，評価を行なうかを説明する．その上で，本機構と比較対照との比較結果を示す．

本機構はフィルタリング機構の管理を目的としており，以下に示す，フィルタリング機構との比較を行なう．

フリーのフィルタリング機構の実装

オープンソースで開発されている，さまざまなフィルタリング機構．どのプロジェクトによるものも，ほぼ同じ機能を備えているので，一括して比較する．

Cisco Secure Policy Manager

Cisco 製品のセキュリティポリシー管理機構

本機構と比較対照との比較は，以下の三つの項目について行なう．

他のフィルタリング機構のルールとの互換性

他の実装によるフィルタリングルールを，利用できるかどうかを評価する．大規模ネットワークでは，それぞれのネットワークにおいて必要とされる機能が異なるため，機材もそれに応じて配置しなければならない．しかし，ネットワーク全体でセキュリティポリシーを同一の記述言語で表記したい場合，他のフィルタリング機構との記述方式互換性がなければ利用できない．

他のフィルタリング機構のルールとの互換性を比較することで，汎用的なセキュリティポリシー記述言語の有用性を評価する．

セキュリティポリシーの配布機構

フィルタリング機構として，セキュリティポリシーであるフィルタリングルールを，ネットワーク上で共有・配布する機構があるかどうかを評価する．汎用的なフィルタ記述言語あったとしても，それをネットワーク上で配布する機構がなければ，管理者が一台一台にセキュリティポリシーをインストールしなければならず，非常に管理コストが高くなる．

セキュリティポリシーの配布機構の有無を比較することで，分散環境におけるセキュリティポリシー配布機構の有用性を評価する．

ログ管理機構

パケットフィルタリングをする場合，フィルタリングの実装が終わった後に継続的に行なわれる，監視も非常に重要な機能である．セキュリティ技術や攻撃手法は日々進化しており，パケットフィルタリングも一度実装が終われば，放置してよいというものではなく，日常的に監視を行なう必要がある．

ログ管理機構の内容を比較することで，集中的なログ管理機構の有用性を評価する．

本機構と比較対照との比較結果を，表 5.2に示す．

表 5.2: 本機構と他のフィルタリング機構との比較

	他のフィルタリングルールとの互換性	ポリシー配布機構	ログ管理機構
フリーのフィルタリング機構	×	×	ローカル
Cisco Secure Policy Manager	×	○	ネットワーク全体
本機構	○	○	ネットワーク全体

項目 1 に関して比較すると，フリーのフィルタリング機構と Cisco Secure Policy Manager は，どちらも独自のフィルタリングルールを持っており，他のフィルタリング機構のルールとの互換性はない．したがって，ネットワーク全体で，統一的なフィルタリングルールの表記を目的とする場合，本機構が唯一の選択肢となる．したがって，本機構は他のフィルタリング機構・管理機構よりも有用性が高い．

項目 2 に関して比較すると，Cisco Secure Policy Manager と本機構だけがセキュリティポリシーの配布機構を持っている．本機構はセキュリティポリシーの配布に HTTP を利用しており，ポリシーの配布において問題が発生した場合，問題の原因を追求するのが簡単になっている．

項目 3 に関して比較すると，フリーのフィルタリング機構はローカルにしかログを保存することができない．syslog などの機能をつかえばリモートにログを送信することはできるが，syslog 自体安全性を考慮していないプロトコルであるため，ネットワーク上で安全にログを送信するのは困難である．本機構はログの収集にも SSL 上での HTTP を利用しており，安全にログを収集することが可能である．

以上に示したとおり，本研究で提案して統一的なセキュリティポリシー管理機構は，現状で利用可能である他のフィルタリング機構，セキュリティポリシー管理機構よりも有用であることを示せた．

第6章　結論

本章では，本研究の結論として，まとめと今後の課題について述べる．

6.1　まとめ

本研究では，ネットワークにおけるセキュリティポリシーを統一的に管理する機構を提案し，実現した．これによって，ネットワークセキュリティの管理コストを低減することができ，安全なネットワークの構築・運用を容易にできるようになった．

ネットワークで行なわれているセキュリティ対策を，その目的によって分類し，現在行なわれているネットワークセキュリティについて，分類にしたがって言及した．その上で，それぞれの分野において基礎的な安全性を実現するのに用いられているパケットフィルタリング技術を，その目的によらず統一的に管理できることを述べた．

また，パケットフィルタリングを大規模ネットワークで管理・運用する上での問題点とパケットフィルタリングルールをセキュリティポリシーと定義することによって，ネットワークにおけるセキュリティポリシー管理を一元化できることを指摘し，統一的な管理機構を実現するのに必要な要件について論じた．

インターネットの利用者が増加し，インターネットが一般の重要な通信インフラとして利用されるようになった．企業や家庭にも広く普及し，インターネットを通じた商取引も，企業間・企業個人間を問わず，活発に行なわれている．

その一方，インターネットプロトコルがその成り立ちから，セキュリティの機能を持っていないことに乗じて，さまざまな攻撃がインターネット上で行なわれるようになった．攻撃の内容はさまざまであり，それに対する防御策も多岐に渡っている．

攻撃に対する防御策として，現在最も一般的に使われているのは，パケットフィルタリング技術である．パケットフィルタリング技術は，ファイアウォールや IPsec などに使われる，基礎的な技術であり，インターネットにおける基本的な安全性を提供する．

しかし，大規模ネットワークにおいて複数のパケットフィルタリング機構を運用する上で，パケットフィルタリングルールの非統一，セキュリティポリシーの配布機構の欠落などの，統一的なセキュリティポリシー管理機構の欠如が問題となっている．

本機構は，これらの問題を解決するため，汎用的なセキュリティポリシー記述言語，セキュリティポリシーリポジトリ，リポジトリとパケットフィルタリング機構間においてセキュリティポリシーの配布を行なうポリシー配布エージェント，そして各パケットフィルタリング機構からの情報を収集する機構のそれぞれを提案し，統一的なセキュリティポリシー管理機構として実現した．

また，ポリシー配布エージェントによって，本機構の汎用的なセキュリティポリシー記述言語と既存のシステムのフィルタリングルールとの変換を行うことで，本システムと既存のシステムとの親和性を高めた．

この結果，既存のシステムに本システムを組み込むことで，大規模ネットワークにおけるパケッ

トフィルタリング機構の管理・運用コストを低減し，安全なネットワークをより容易に構築・運用できることを示した．

6.2 今後の課題

今後の課題としては，以下の三つが挙げられる．

- 既存のパケットフィルタリング機構との親和性の向上
- より一般的なフィルタリング機構への対応
- 他のセキュリティ技術も含めた統一的な管理

本節では，それぞれの今後の課題について述べる．

6.2.1 既存のパケットフィルタリング機構との親和性の向上

本研究では，既存のパケットフィルタリング機構との親和性を保障するために，ポリシー配布エージェントによって，本研究の定義した汎用的なセキュリティポリシー記述言語と，既存の機構で使われているフィルタリングルールとの相互変換を行なった．

しかし，既存のパケットフィルタリング機構には，本研究で対象とした機構の他にも多数あり，それらのフィルタリングルールと，本機構の記述言語との互換性はない．本機構を実際のネットワークで利用するためには，より多くのフィルタリング機構との互換性が望まれる．

また，本研究が対象としているフィルタリング機構のフィルタリングルールとの互換性も完全ではない．既存のフィルタリング機構のフィルタリングルールは，オプションのルールであるほど，独自の記述方式をとってることが多く，それらの対応は本研究の対象外とした．

しかし，本機構を実用的に利用するためには，それらのオプションへの対応も必要となる．

6.2.2 より一般的なフィルタリング機構への対応

本研究では，セキュリティポリシーの統一的な管理を目的としていたため，セキュリティ技術と関連性のない，他のパケットフィルタリング技術は対象としなかった．

しかし，現在のインターネットではその他にもパケットフィルタリング技術は使われている．たとえば，QoS 技術もその一つである．QoS では特定のアドレス・ポート番号などをもった一連のデータフローを一まとまりのものとして扱い，class に応じてキューイングなどを行なっている．

フィルタリングルールによってパケットを区別化し，その区別化されたデータフローに対して何らかの処理を行なうという点で，セキュリティの向上を目的としたパケットフィルタリングと同じ構造を持っている．実際に FreeBSD では，ファイアウォールと帯域制御技術を同じインターフェースで管理することができるようになっている．

したがって，セキュリティポリシーリポジトリを，QoS におけるフィルタリングルール・処理とともに使える形式に拡張し，一元的に管理することが可能であると考える．

これによって，セキュリティポリシーの管理を，ネットワークにおけるパケットフィルタリング一般の管理へと拡張することができると思われる．

6.2.3 他のセキュリティ技術の統一的な管理

6.2.2節では，本機構をパケットフィルタリングの統一的管理機構としてより一般化する方向性を示したが，その他に，セキュリティ技術の統一的管理機構として一般化する方向性もある．

たとえば，ネットワークベースで現在広く利用されつつあるセキュリティ技術として NIDS(Network-based Intrusion Detection System) が挙げられる．

NIDS はネットワークのトラフィックをすべて監視し，特定のシグネチャ(トラフィックパターン)に適合するデータを発見すると，管理者にアラートを送信するシステムである．NIDS の運用上の問題点としてシグネチャの管理が挙げられる．NIDS は基本的にはシグネチャを持っていない攻撃パターンを検知することはできないので，常に最新のシグネチャを保持しなければならない．これは，NIDS が増えるにしたがって管理コストが増えることを意味している．

NIDS はパケットをフィルタしてはいるが，本研究の対象としたパケットフィルタリング機構のようにトラフィックを選別して処理を行なうというシステムではない．

しかし，NIDS のシグネチャは基本的に，プロトコルタイプ，サービス (ポート番号)，ペイロードの内容などで構成されており，フィルタリングルールと近い構成をしている．そこで，6.2.2節で述べたのと同じように，本研究の対象としてセキュリティリポジトリを拡張することで，NIDS のシグネチャの管理を，本機構の一部として行なうことができると考える．

ファイアウォールと NIDS を合わせて管理することで，相補的な効果を期待することができる．

謝辞

本研究を進めるにあたり，暖かい御指導を頂いた，主査である慶應義塾大学環境情報学部教授村井純博士に感謝します．また，丁寧な指導を頂いた副査である慶應義塾大学環境情報学部助教授中村修博士と東京大学大学院助教授江崎浩博士に感謝します．

日頃から絶えず指導して下さった，慶應義塾大学環境情報学部助教授楠本博之博士，慶應義塾大学環境情報学部専任講師の南政樹氏，慶應義塾大学政策・メディア研究科の関谷勇司氏，小原泰弘氏には特に感謝します．皆様のご指導がなければ決して修了することはできなかったでしょう．

論文執筆に際して，多大なる支援をいただいた，慶應義塾大学政策・メディア研究科の佐藤雅明氏，安田歩氏，慶應義塾大学環境情報学部の三屋光史朗氏，小畠元氏，渡里雅史氏，慶應義塾大学総合政策学部の日野哲志氏，小柴晋氏，そして修論準備委員会の皆様に感謝します．こうして無事，修士論文を書き終えられたのは彼らの協力によるところが大きいです．あ，あと海崎良氏にも．

そして，様々な激励と助言を頂いた，WIDE プロジェクト IPv6 WG の皆様，慶應義塾大学村井・徳田・楠本・中村・南研究会の SING の皆様と研究会の諸氏に感謝します．

最後に，その比類なき歌声で私に力をくれた，”**うたうたい**” 矢野真紀氏に感謝と畏敬の念を表します．

参考文献

[1] Check Point Software Technologies Ltd. http://www.checkpoint.co.jp/products/firewall-1/fw1gui.html, January 2002.

[2] 電子商取引推進協議会. 「平成１２年度電子商取引に関する市場規模・実態調査」, January 2001.

[3] T. Dierks and C. Allen. The TLS Protocol Version 1.0. Request for Comments (Proposed Standard) 2246, Internet Engineering Task Force, January 1999.

[4] M. Elkins, D. Del Torto, R. Levien, and T. Roessler. MIME Security with OpenPGP. Request for Comments (Standard Track) 3156, Internet Engineering Task Force, August 2001.

[5] B. Ramsdell and Ed. S/MIME Version 3 Message Specification. Request for Comments (Proposed Standard) 2633, Internet Engineering Task Force, June 1999.

[6] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. SSH protocol architecture (work in progress). Internet Draft, Internet Engineering Task Force, November 2001.

[7] J. Myers and M. Rose. Post Office Protocol - Version 3. Request for Comments (Standard) 1939, Internet Engineering Task Force, May 1996.

[8] C. Newman. Using TLS with IMAP, POP3 and ACAP. Request for Comments (Proposed Standard) 2595, Internet Engineering Task Force, June 1999.

[9] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force, November 1998.

[10] 情報処理振興事業協会 セキュリティセンター. 「情報セキュリティの現状 2001 年版 (ドラフト 0.1)」, November 2001.

[11] FreeBSD Project. http://www.freebsd.org/, January 2002.

[12] NetBSD Project. http://www.netbsd.org/, January 2002.

[13] The Linux Kernel Archives. http://www.kernel.org/, January 2002.

[14] Cisco Systems Inc. http://www.cisco.com/, January 2002.

[15] Darren Reed. http://coombs.anu.edu.au/ avalon/ip-filter.html, January 2002.

[16] netfilter. http://netfilter.samba.org/, January 2002.

[17] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). Request for Comments (Proposed Standard) 2406, Internet Engineering Task Force, November 1998.

[18] S. Kent and R. Atkinson. IP Authentication Header. Request for Comments (Proposed Standard) 2402, Internet Engineering Task Force, November 1998.

[19] KAME Project. http://www.kame.net/, January 2002.

[20] Linux FreeS/WAN. http://www.freeswan.org/, January 2002.

[21] C. Christiansen, N. Freedman, and B. Burke. *Internet Security Software Market Forecast and Analysis, 2000-2004*. IDC, October 2000.