JAIST Repository
https://dspace.jaist.ac.jp/

Title	ネットワークセキュリティの統合的管理手法に関する研究
Author(s)	武智，洋
Citation	
Issue Date	2000-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1327
Rights	
Description	Supervisor:篠田 陽一，情報科学研究科，修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修 士 論 文

ネットワークセキュリティの統合的管理手法に関する研究

指導教官　篠田陽一 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

武智 洋

2000 年 2 月 15 日

Copyright © 2000 by Hiroshi Takechi

要 旨

ネットワークの急速な普及とそれに伴って増大しているクラッカーやコンピュータウィルスなどの脅威によって、ネットワークのセキュリティ設定を維持管理することは非常に重要になってきている。そのため、企業や大学など一般の組織では、組織の運営方針を反映したセキュリティポリシーを設け、ネットワーク構築をおこなうようになってきている。

一方、ルータのアクセスコントロールリストの設定や各種ホストのログイン設定は、通常、管理者が各機器にログインしてコマンドインターフェイスによりおこなうのが一般的であり、このような状況では、管理者のセキュリティポリシーの解釈によって各種設定が決まり、実際のセキュリティに関する設定が、セキュリティポリシーを順守しているかの検証はできない。また、大規模なネットワークでは、全体として一貫性を持つように維持管理することは不可能となってくる。

このように、最終的な設定が人手によっておこなわれるのは、

- セキュリティポリシーと具体的な機器設定との抽象レベル差が大きい。
- 管理ドメイン、ネットワークトポロジーなど各構成項目が、多岐にわたると共に相互に関連している。

といった要因が考えられ、また、ネットワークトポロジや組織構成の変更などが頻繁に起ることが、人手での管理をさらに困難にしている。

そこで、本研究では、ポリシーに基づいたネットワークの一元管理システムの枠組みを考え、特に各種ネットワーク上のサービスを記述したポリシーからネットワーク上のアクセス制御機構を持つ機器でどのようなアクセス制御を行うべきかの規則を導出するための手法とネットワーク上でポリシー記述の要件を満たしていない機器を特定するための手法を提案し、プロトタイプを設計/実装して有用性を確認した。また、ポリシーによるセキュリティ管理に関連した各種システムの調査を行い、ポリシーを基にしたセキュリティ管理システムにおいてどのような点が問題となるのかについて議論し、今後の展望を考察した。

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1 背景

ネットワークが非常な勢いで普及しつつある中で、ネットワーク管理においてセキュリティ管理を如何に行うかが大きな問題になりつつある。

ネットワークの普及に伴って、クラッカーやコンピュータウイルスなどの脅威も急速に増えつつあり、それらに常に迅速に対応し、ネットワークの各種設定を維持管理していくことは多くの組織において管理者の負担を増大させている。

また、組織のネットワークにおいては、その組織の目的を達成するためにネットワークを利用できるように運用管理していかなければならない。そのために、多くの組織では、コンピュータシステムおよびネットワーク運用のためのポリシーを定めている。ネットワークは、この組織のポリシーに基づいて運用管理されてなければならない。しかしながら、多くの場合、組織のポリシーは非常に抽象度の高いレベルで記述されており、実際にネットワーク運用に反映するためには、それを解釈し、ネットワークの機器等の設定に反映できるようなより具体的な低いレベルの手続きに変更することが必要になる。

企業や大学など一般組織においては、管理者が経験に基づいて、ポリシーの解釈および各種ネットワーク上の機器設定などを行っていた。一部ネットワーク管理システムでは、管理コンソールから複数の機器設定を行えるものも出てきているが、設定できる機器はある特定ベンダーの機器だけに限定されたり、設定できる機能が一般的なインターフェイスにおいて可能なものだけの制限などから、それによって運用管理がすべて行えるわけではないのが現状である。

複数ベンダー、様々な種類のハードウェア・ソフトウェアによって構成されるネットワークにおいてすべてのセキュリティ設定を手動で行うことは、非常優秀な管理者であっても難しいタスクである。しかも、ネットワーク管理は日々刻々改善していく必要のあるプロセスであり、有能な管理者を日々の運用にあてることは多くの組織では実質上不可能である。

ポリシーに基づいたネットワーク管理が容易に行えるような管理手法が必要となってきている。

1.2 本論文の目的

そこで、本研究では、ポリシーを基にしたネットワークの一元管理システムの枠組みを考え、特に以下のような各種ネットワーク上のサービスを記述したポリシーからネットワーク上のアクセス制御機構を持つ機器でどのようなアクセス制御を行うべきかの規則の導出と、ネットワーク上でポリシー記述の要件を満たしていない機器の検出を行う手法を提案し、プロトタイプを設計/実装して有用性を確認する。

1. ネットワークをサービスの集まりとして考え、提供したいサービスを定義する。
2. サービスの定義は以下のような要素で構成されるポリシー記述によって表現する。

   オブジェクト – サービスを提供するネットワーク上の資源

   サブジェクト – サービスを利用するユーザ等

   アクション – サブジェクトのオブジェクトに対する動作

   制約 – ポリシーが適用される際の条件
3. ポリシーのサブジェクトとオブジェクトからサービスをネットワーク上にサービス属性としてマッピングする。
4. ネットワーク上で属性計算を行うことでサービスのネットワーク上での広がりを求める。
5. ポリシーから求めた各属性間の関係を定義した属性評価式に基づいて、各属性を評価することによって、アクセス制御規則設定がポリシーに違反していないかの検出を行う。

6. また、アクセス制御点において、属性評価式を満足するようなアクセス制御規則を求めることで、各ネットワーク上の機器のアクセス制御規則の導出を行う。

また、ポリシーによるセキュリティ管理に関連した各種システムの調査を行い、ポリシーを基にしたセキュリティ管理システムにおいてどのような点が問題となるのかについて議論し、今後の展望を考察する。

1.3 構成について

第1章では、本論文の背景および目的を述べ、本論文の構成を紹介する。第2章では、ネットワーク管理の現状と問題点について言及し、それに対してどのようにアプローチすべきかを述べる。また、ポリシーに基づいたネットワーク管理に関しての一般的なモデルについて説明する。第3章では、各種関連研究がどのような部分を対象としているかとその問題点について論じ、そこから、今回の提案する手法の位置付けを明確にする。第4章では、提案手法について解説する。第5章では、プロトタイプについての解説と問題点について述べる。第6章では、提案手法およびポリシーに基づいたネットワーク管理における課題について述べる。第7章で、本研究のまとめを述べる。

第 2 章

ネットワークセキュリティ管理の問題点と現状

2.1 問題点

現状のネットワークセキュリティ管理の問題点として以下のようなものがあげられる。

- ネットワークの急激な普及と各種脅威の台頭
- 優秀な管理者の不足 – 管理者の技術レベルへの要求がますます高くなる
- ポリシーと機器設定の抽象化のギャップが大きい
- ポリシー変換の有効な手段の不足
- ネットワークは常に改善されるプロセス

新しき技術の導入やユーザの増加などどこの組織においても管理者は次々とネットワークを維持管理するための対応をし続けなければならない状況に陥っている。このため、組織におけるネットワークセキュリティは、あらかじめ定められたセキュリティポリシーに基づいて実際のネットワーク機器のアクセスコントロールなどの設定を行うなってきている。しかし、現在の多くの組織では、ポリシーから実際の設定へのマッピングは、自動的に行われるわけではなく、ネットワーク管理者がポリシーを解釈し、それに合致するような各機器独自の設定を考え出す方法が一般的に用いられている。その理由としては、

- 一般にポリシー記述は人間が理解運用する形式で記述され、抽象度の高い内容のものが多い
- 手続き的な内容のポリシーであっても自然言語によって記述されているため設定を自動生成できる形式になっていないことが多い

ことが挙げられる。

このような、管理者の解釈による実際の設定に関する問題点として、

- 解釈の間違いや、設定への変換ミスにより設定に間違いが生じやすい
- 複数の機器に跨った設定を必要とする場合など、各機器間の設定の整合性を保つのが非常に難しい

ことなどが挙げられる。

さらに、ネットワークは一度構築すれば終了するものではなく、

- 人の出入り
- 新しいサービスの導入
- 新しい機器の導入やリプレースによるネットワーク構成の変化
- セキュリティホールの発見などの外的要因

などの状況変化に対応して、各種構成・設定を修正していく必要がある。

各種構成・設定を修正を行う作業において、現状では、各種機器の設定とその設定に対応するポリシーとの間に関連付けがないため、修正に対応するポリシーも同時に修正することは、非常に困難になりがちである。

結局、ポリシーと実際の設定の間にずれが生じることとなり、設定の意味を理解するのが徐々に難しくなるため、さらに、設定変更に時間がかかるなどの問題が生じる。

2.2 アプローチ

このような問題点を回避するためには、状況変化に対してアドホックな対応によってネットワークを構築するのではなく、ネットワークの行うべき事を定めてポリシーとし

て記述し、常にポリシーにそった設定によって運用を行い、状況変化に対してはポリシーを変更することで対応することで対処しなければならない。

また、状況変化に対してすばやく対応するには、ポリシーから実際の運用管理オペレーションへのマッピングが容易にできなければならない。

そのため、高い抽象度のポリシーを実際のオペレーションの手続きまで人手を介さず自動的に生成できることが望ましい。また、実際のネットワークがポリシーに準拠しているのかどうかを検出できることがポリシーと実際の設定との整合性を維持するのに必要である。このように、ポリシーに基づいたネットワーク運用が行えるための枠組みと手法を確立することが重要である。

2.3 ポリシーに基づくネットワーク管理の一般的モデル

ここでは、ポリシーに基づくネットワーク管理の概要を説明する。

2.3.1 ポリシーシステムの概要

ポリシーシステムの目的は、全体としてネットワークを制御管理し、そのネットワークを運用している組織の目的を達成するためのネットワーク運用管理おこなうことである。究極的には、そのような制御を達成するためには、ネットワークを構成する個々のエンティティの振る舞いを変更できることが必要である。そのため、全体としてネットワークに対するポリシーを設定し、ネットワークポリシーを通じてネットワーク上のエンティティの振る舞いを制御する。

このポリシーシステムの枠組みとして IETF の Policy Framework ワーキンググループで議論されているモデルを紹介する。

ポリシーシステムでは、ポリシールールをもとに構築されるシステムであり、以下のようなことが行えなければならない。

- ユーザがポリシールールを定義し更新できる
- ポリシールールを保存格納および読み出すことができる
- ポリシールールを解釈し、実施、強制できる

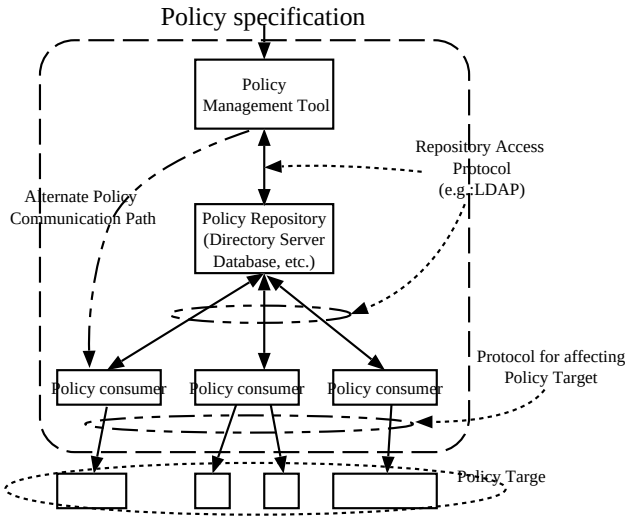

図 2.1: ポリシーフレームワーク WG のモデル

これらを実現するためにいくつかの機能ブロックからポリシーシステムは構成される。(図 2.1)

ポリシー管理ツール :

ユーザやアプリケーションがポリシールールを定義し更新するためのツール。また、ポリシールールがちゃんと機器に適用されているかのモニタリングを行う。GUI あるいはコマンドラインのインターフェイスを持ち、主な機能としては、

- ポリシー編集
- ポリシー表現
- 規則の変換
- 規則の検証
- 全体的なルール間の不整合検出

などを行う。この時点で、ポリシーが人が読める形式から、リポジトリーに格納する情報モデルの形式に変更される。また、ポリシー間のコンフリクト検出も一部行われる。

ポリシーリポジトリ :

ポリシールールを保存しておくためのリポジトリーであり、機能としては、

- ルール格納
- ルール検索

などを行う。

ポリシーコンシューマ :

ポリシールールを獲得し、解釈して、ポリシーターゲットが利用できる形に変換を行って、実際に適用を行う。機能としては、

- ポリシールールのポリシーターゲットへの設定
- ポリシーターゲットであるデバイス固有の処理への対応
- ネットワークと資源に対するポリシールールからの要件の検証
- ポリシールールの変換

を行う。

ポリシーターゲット :

ポリシールールに記述された内容を実行するエレメントであり、ポリシーを実行するのに加え、

- ポリシールールの実行時検証
- ポリシールール間のコンフリクトの検出

などを行う場合もある。

まず、人が理解できる形式でのポリシーがポリシー管理ツールによって編集され、管理者は、実際のサービスとして行いたいことを表現したポリシーを生成する。これは、管理ツール上で管理者が人手で行う作業になる。ポリシーを個々のサービスを記述したポリシーに分割し、ポリシーの情報モデルにあった形式に変換を行う作業である。

その後、ポリシー管理ツール上でポリシールール不整合や文法エラーなどを検査した後、ポリシールールはポリシーリポジトリーに格納される。

ポリシーリポジトリーに格納されたポリシールールは適宜ポリシーコンシューマによって検索され読み出される。

ポリシーコンシューマーでは、デバイス独立に記述されたポリシールールをさらに実際のデバイスに設定実行できる形式に変換して、ポリシーターゲットに適用する。

ポリシーターゲットでは、設定されたポリシーあるいはパラメータにしたがって、フィルタリング等の動作を行う。

ポリシーコンシューマとポリシーターゲットは論理的なエンティティであり、実際の機器では、様々な組合わせや実装が考えられる。通常、ポリシーリポジトリを介してポリシー管理ツールとポリシーコンシューマの間でポリシールールが受け渡しされるが、別のパスとしてポリシーリポジトリーを介さず直接受け渡しされる場合も考えられる。

2.3.2 ポリシー階層レベル

ポリシーは、記述内容によって階層構造をしていると考えられる。ポリシーがいくつかの階層から構成されるべきであるかは、様々な意見があり定まっていないが、本論文では、以下の４つの階層があると考える。

1. 組織のポリシー
   まず、最初は組織の目的や目標を表現したポリシーで一般に人が解釈し理解するために自然言語によって表現される。組織の命令階層などを色濃く反映し、抽象度の高い表現が行なわれているために、実際にネットワーク上にポリシーで表現された目的や目標を反映するためには、ネットワーク上の機器で扱える形への変換が必要になる。
2. SLA(Service Level Agreement)
   上記ポリシーからネットワーク上でどのようなことが行なわれなければならないかを記述したポリシーであり、サービスプロバイダとの契約などのレベルの記述と考えられる。
3. SLO(Service Level Objective)
   SLA を実現する各種サービスについて記述したポリシーであり、これらは組織の目的を実現するために提供されるネットワーク上の具体的な各種サービスをポリシー

として述べたものである。実際のネットワーク構成や各種機器の機種などと独立した形での記述が行われる。

4. デバイスレベルの記述
   最後のレベルとしては、実際にネットワーク上の機器動作制御を行うために、機器が解釈し実行できる表現の記述が行われる。

2.3.3 ポリシー記述

ポリシー記述として様々なものが提案されているが、基本的な記述方法として数多く使用されるオブジェクトとしてポリシーを記述する方法を説明する。ポリシーは以下の要素から構成され、

- 様相
- サブジェクト
- ターゲットオブジェクト
- ゴール
- 制約

『ある**制約**の中で **オブジェクト**に対して **サブジェクト**が何らかの操作（**ゴール**）を行う事に 対して許可または禁止などする。』

という記述が行われる。

ポリシーの様相 (modality)

ポリシーは、何らかの行動を起こそうとする意思を示したポリシーか、ある行動を行うことができる権限を示すポリシーかの 2 つにわけることができ、ポリシーの modality(様相) という。

前者が “動機 (motivation)” で、後者が “権限委譲 (authorization)” (“権力 (power)”) [1]である。

アクション は “動機 (motivation)” と “権力 (power)” を持つ者が実行することができ、権力を得る方法としては、“権力者 (authority)“からの” 委譲 (delegation)“による。

様相には、4 種類ある。

- Positive authorization (permitting) – 許可
- Negative authorization (prohibiting) – 禁止
- Positive motivation (requiring) – 要求
- Negative motivation (deterring) – 阻止

ポリシーサブジェクトとターゲットオブジェクト

サブジェクトとは、ポリシー制約の範囲内で、ポリシーゴールを実行することを承認されて (authorized)、動機付けされている (motivated) 人を表す。あるコマンドによってあるポリシーが自動化されている場合には、そのコマンドを入力した人がサブジェクトと考えられる。

また、ターゲットオブジェクトとは、ポリシーの対象となっているオブジェクトの集合である。

ポリシーゴール

ハイレベルなゴールかアクションのどちらかを定義しており、あるゴールがどちらであるかは、述べられているポリシーがどういうレベルであるのかによる。オペレーションだけで表現されているゴールがアクションである。マニュアルなどで良く使われる言葉として “Procedure”(手続き) があるが、これは、アクションの連なりのことである。

[1]通常、組織階層の順序付けがしっかり管理されている組織を考えると、権限委譲 (authorization) されていない権力 (power) は考えにくいため、権力を持つことは権限委譲されていると考える。

ポリシー制限

ポリシーオブジェクトに対して、それが有効である条件を示す。時間、期間、その他の条件があるが、例えば、“経理データベースにアクセスできるのは、就業時間中に制限する” などである。

第 3 章

関連研究

3.1 関連研究事例

3.1.1 FAM(Flexible authorization Manager) アーキテクチャ[3]

Jajoia らの主張では、多くのシステムの Authorization model は、そのシステムのセキュリティ機構を反映したものが多く、システム内では一つの特定のポリシーのみ適用されることが多い。実際にはシステム内で様々なアクセスコントロールへの要件があり、それら全てを一つのポリシーで満足させることはできない。そこで、彼らが提案している FAM(Flexible authorization Manager) のアーキテクチャでは、表現能力の高い ASL(Authoriztion Specification Language) でアクセスコントロールポリシーを記述し、一つのシステム内で複数のアクセスコントロールポリシーを利用できるようにし、それに基づいて全てのアクセスコントロールを行うことを提案している。

但し、実際のオブジェクトへのアクセスは FAM Authorization Request Processor というリファレンスモニタを介して行われることを前提にしている。システム全体が FAM のアーキテクチャに統一されるされなければ、アクセスポリシーを強制できず、既存の様々の機器が混在するシステム上で利用するのは現実的でない。事実、関連する論文ではデータベースシステムへの適用に関するものである。

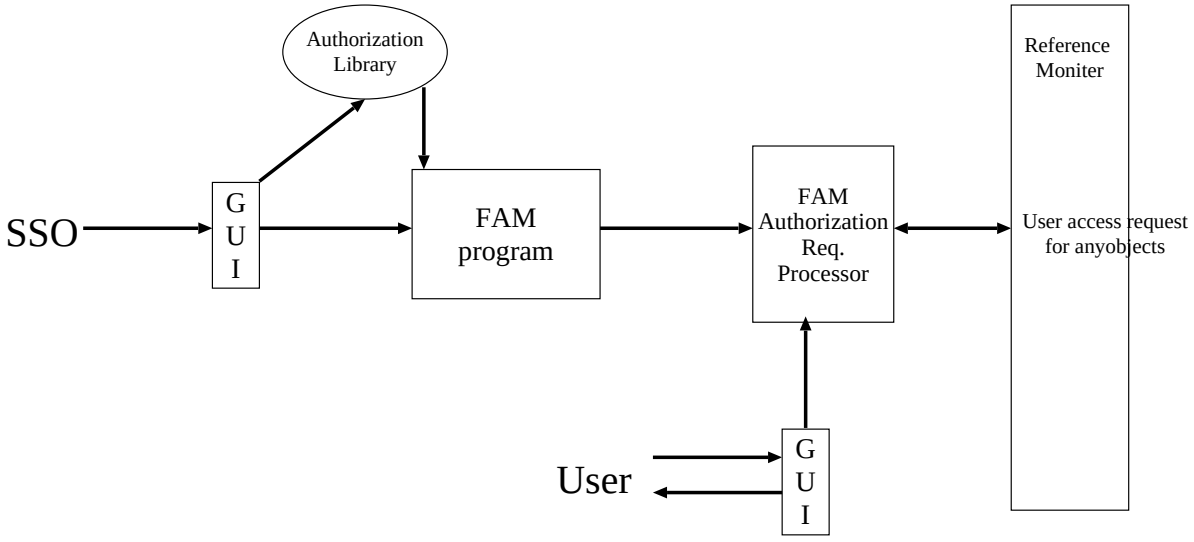

図 3.1: FAM アーキテクチャ

3.1.2 Argos[4]

CHASSIS(Configurable Heterogeneous And Safe, Secure Information System) プロジェクトにおいて、異種データベースが連携したものに対して全体的な層としてアクセス制御システムを提供するために開発された Argos においては、対象とするシステムは FDBMS(Federated database management system) である。やはり、ポリシーを強制するポイントしては、リファレンスモニターを用いている。

3.1.3 Management Policy Service for Distributed Systems[6]

マネージメントポリシーの解釈を自動的に行うことができれば、ポリシーを変更するだけで、分散管理システムの挙動を動的に変更することができると考え、マネージメントポリシーを “authorisation policies”(管理者がどのような活動を許すかについて記述したポリシー) と”obligation policies”(管理者が行わなければならない活動について記述したポリシー) を記述しているものとして、グラフィカルなポリシーエディターによって扱えるシステムを提案している。ポリシーエディターは、抽象度の高いポリシーを実行可能なレベルのより具体的なポリシーに変更する作業や派生したポリシーの管理、ポリシー間の依存関係などを管理するのに用いられる。

3.1.4 Specification and Verification of Security Poilcies[15]

情報システムがセキュリティに関する要件を満たしているかを検証するために、仕様記述言語を用いたフレームワークを提案している。

提案されているフレームワークでは、まず、検証の対象となる情報システムを状態遷移機械で記述できるとして仕様記述言語“Z”でモデル化し、セキュリティポリシーは、情報システムが生成するトレース (状態連鎖) 集合を満足させる一時的な特性として考え、temporal logic に基づいた言語を開発してモデル化し、検証は、一時的な性質が情報システムによって生成されるトレース集合によって充足されるかどうかを証明するプロセスとしてモデル化される。

対象として想定しているのがある一つの情報システムのセキュリティ機構であり、システムのセキュリティ仕組みそのものがセキュリティの要件に合致しているかどうかを検証することである。実際、情報システムの例として記述されているのは、Unix ファイルシステムのプロテクション機構である。

ネットワーク全体を一つのシステムと見て多機種にわたる様々なアクセス制御機構をモデル化することについては述べられていない。

3.1.5 Policy Framework ワーキンググループ

IETF Policy-Framework ワーキンググループでは、ポリシーベースのネットワーク管理を行なうためのフレームワークについて議論が行なわれている。目的とするところは、ポリシーをリポジトリーに格納し、それらをシステム全体で参照、解釈、実行するポリシーシステムにおいて、ポリシーを解釈し、システムにポリシーを適用するポリシーコンシューマ機能ブロックとポリシーによって命じられた操作を実行するポリシーターゲット間での相互運用性を維持することにある。現在、ポリシーシステムの記述の中で使用される管理されるオブジェクト（例えば、ユーザ、ネットワークデバイス、サービスなど）の構造的性質やデバイス間の関係などの情報モデルを特定のプラットフォームへの依存や特定の技術から独立になるようなオブジェクトモデル [14] として規定しようとしている。また、各種ポリシーの情報を特定のリポジトリーにどのようにマップするかのスキーマについても LDAP を用いた方法 [18] について規定されつつある。

3.2 各研究事例からの問題点

各研究事例から以下のような問題点が挙げられる。

3.2.1 ポリシー強制点への仮定

リファレンスモニターを仮定すれば、ポリシー記述で対象となるオブジェクトに対する全てのアクセスにおいて、ポリシーを強制することが非常に容易になる。しかし、ネットワーク管理におけるアクセス制御ポリシーは対象となるオブジェクトに対してのアクセスを横取りすることが可能となるアクセスポイントは得られず、リファレンスモニターを置くことは困難である。

したがって、アクセス制御ポリシーを強制するポイントが異種機器および異種レイヤに分散していることを前提として、それらを如何に一つのアクセス制御ポリシに従った設定に維持管理できるかが、ネットワークセキュリティ管理を統一的に行うための重要な点である。

3.2.2 ポリシー記述方法

ポリシー記述方法としては、仕様記述言語を用いてシステムとアクセスコントロール機構をモデル化し、そのモデルでポリシーを記述する方法と、システムのセキュリティの対象となるものを”オブジェクト”、オブジェクトに対してアクセスなどするユーザなどを”サブジェクト”、サブジェクトがオブジェクトに対して行う操作を”アクション”として、あるサブジェクトがあるオブジェクトに対して何らかのアクションを行ってよいかどうかを記述する方法とがある。

仕様記述言語を用いる方法の長所としては、記述が正確に出来るならば、セキュリティ機構および設定に関して、自動的な検証を可能である。また、ポリシーから実際の設定への自動的な変換も可能である。

短所としては、管理者がポリシーを正確に記述するためには、仕様記述言語を理解しなければならず、負荷が大きい。また、頻繁にシステムが変更される場合に、モデル化をし直すことになるようであれば、迅速な運用に対応することが難しくなる。

また、仕様記述言語がポリシーを記述するだけの表現力を持つかどうかも問題になってくる。仕様記述言語の表記方法が管理者が内部に持っているポリシーの表現と著しく異なっていて、直感的にポリシーを書き下すことが難しい場合には、ポリシー翻訳で間違いが生じやすい結果となる。

“オブジェクト” – “サブジェクト” – “アクション” を用いる方法では、管理者が考えていることをそのまま素直に表現でき、管理者にとっては扱いやすいといえるが、セキュリティ機構の自動検証などは難しく、また、実際に機器が扱える形のポリシーになるまで、具体的なものに記述していく必要がある。

3.2.3 ポリシー変換について

ポリシーを変換することについては、最終的にどこでポリシーが解釈実行されるかに依存する。ポリシーが既存の機器上で解釈されるような場合には、ポリシーエディタなどで人手で変換を行う形になっている。ポリシーを解釈する部分を限定している場合には、変換をせずそのまま解釈する形になっている。その場合には、リファレンスモニターの導入が必要になってくる。

IETF Policy-Framework WG では、ポリシーを使ったネットワーク管理の枠組みと情報モデルの標準を決め各種ベンダー機器間での相互運用性を確保しようとしているが、全ポリシーがネットワーク全体で目的を達成できているかの整合性チェックなどについては具体的に述べていない。

ポリシー変換については特に確立された手法は提案されていない。

第 4 章

サービス記述よるネットワーク管理手法について

本章では、提案手法について説明を行う。まず、提案するシステムの全体概要を説明した後、個々の手法の詳細について述べる。

4.1 システムの全体構成と処理の流れ

本論文では、先の第 3.2 章で述べたような問題を解決するために図 4.1 に示すような全体構成を持つサービス記述によるネットワーク管理手法の枠組みと、属性計算によりアクセスコントロールポリシーをネットワーク上の実機器上のアクセスコントロール規則への翻訳を行う手法およびポリシー違反を検出する手法を提案する。

4.1.1 サービス記述によるネットワーク表現

まず、管理者はネットワークをどのように管理運営していくかを定義しなければならない。管理者の立場から見ると、ネットワークはある組織の目的を達成するためにユーザが必要なサービスを提供するものであると考えられる。そこで、提案する手法においては、提供したいネットワークをサービスの集合から構成されるものとして、管理者がサービスを記述することによって提供したいネットワークを定義することからスタートする。

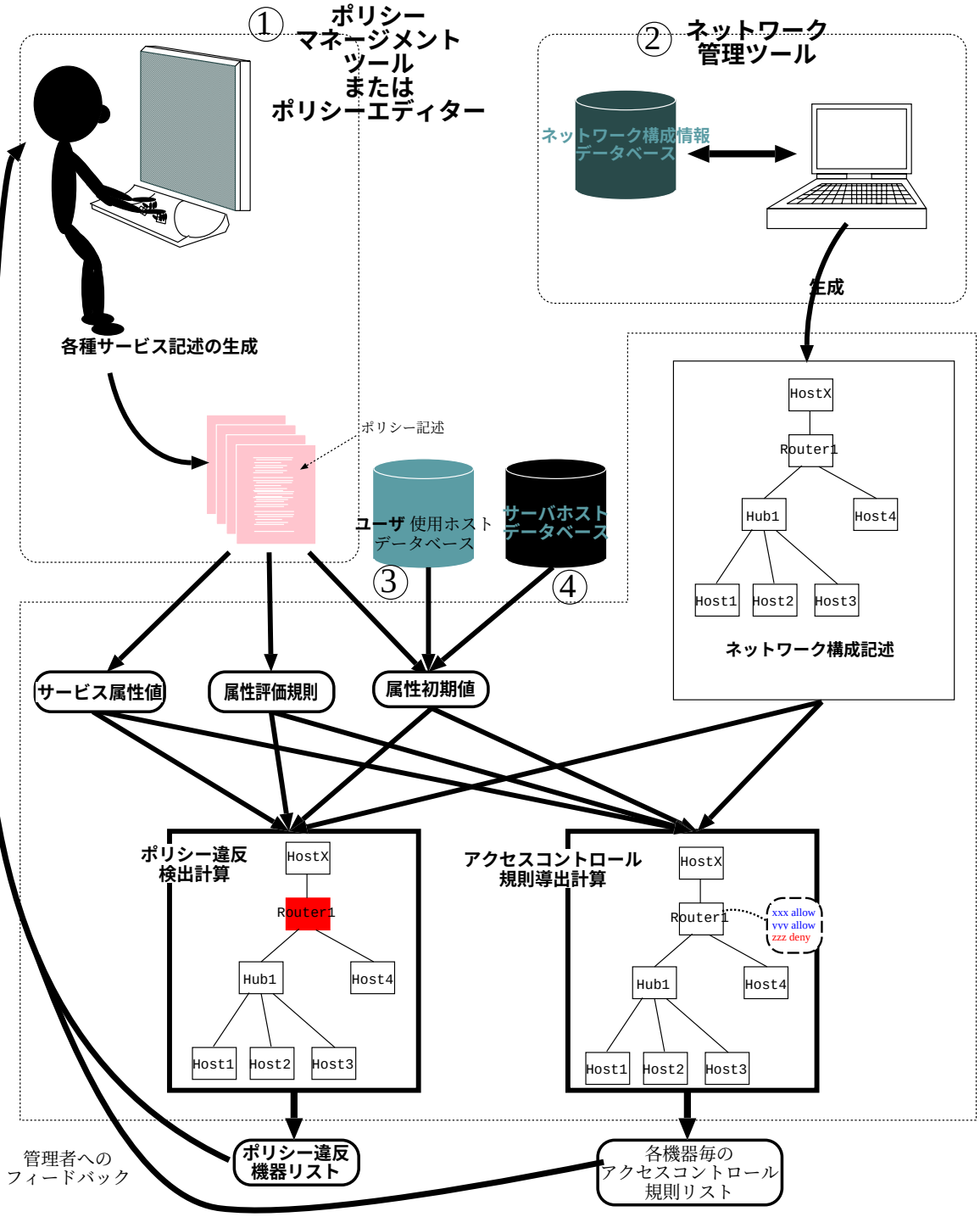

図 4.1: システム全体構成

サービス記述は、サービスを”利用する側”がサービスを”提供する資源”に対して何らかの“動作を行えるかどうか”を記述する。

それぞれを以下のように表現する。

- サービスを提供する資源を”オブジェクト”として表現する。
- ユーザ等のそれを利用する側を”サブジェクト”とする。
- サブジェクトがオブジェクトに対して行う動作を”アクション”として表す。

また、アクションを行う際の条件を”制約”として与えられるようにする。サービス記述はこれらのアクセスコントロールポリシーの組合わせで表現される。

一般組織におけるのポリシーは第 2.3.2 章で述べたように記述の抽象度が異なる幾つかのレベルが考えられる。そこで、本提案では、人が読んで理解することしかできないレベルのポリシーは”ポリシーマネージメントツール”あるいは”ポリシーエディタ”といったツールを用意し (図 4.1 の ①)、サービス記述が可能なポリシー記述にまで人手で翻訳できるようにする。

“ポリシーエディタ”では、以下のような機能を持つ必要がある。

- 関連している抽象度の高いポリシー記述のリンクのデータベース
- 抽象度の高いポリシーとその派生ポリシーの関係を維持管理するデータベース
- 各ポリシーに対する管理者・責任者の維持管理機構
- 変更・追加に伴う諸作業のサポート

セキュリティ管理者およびネットワーク運用管理者は、これらのツールを利用して、構築したいネットワークのサービス記述を定義する。

4.1.2 ポリシー記述から属性計算準備

定義したポリシー記述から属性計算に必要な以下の情報を生成する。

- サービス属性値

- 属性評価規則
- 属性初期値

サービス属性値はサービスに関係しているネットワーク構成上のエンティティに与えられる属性である。ポリシー記述のうち Authorization Policy から生成される。

属性評価規則は属性が満たすべき条件を記述したもので、ポリシーの制約等より生成される。

属性初期値とはネットワーク構成記述上のエンティティに対して属性計算の初期値として与えられるもので、サブジェクトとユーザとホストの対応関係データベース (図 4.1 の③) とオブジェクトとサービス提供ホストのデータベースである”サーバホストデータベース” (図 4.1 の ④) から決定する。

4.1.3 ネットワーク構成記述の生成

ネットワーク構成記述は実際のネットワークの構成をいくつかのレイヤに分割して表現したデータ構造を持っている。物理的なレイヤではネットワーク上のエンティティを物理的配線状態で辿ることができようなネットワーク構成を表現し、論理的ネットワークレイヤでは VLAN など仮想ネットワークでのネットワーク構成を表現する。

また、各レイヤ毎に必要な属性を持っており、例えば、物理的なレイヤでは、機器の設置場所が物理的に安全であるかの情報を提供する属性を持つ。属性計算ではこれらのこれらの属性も評価のパラメータとして用いる。

アプリケーションから物理層までのネットワークを構成するエンティティの情報が必要である。ネットワークの通常の構成情報はネットワーク管理ツールを利用し生成し、それにネットワーク構成記述に必要な属性や各レイヤ間の接続関係記述を追加して生成する。

4.1.4 ネットワーク構成記述上の属性計算

まず、属性初期値から、サブジェクトと対応するネットワーク構成記述上のエンティティにサービス属性を与える。また、オブジェクトと対応するエンティティにも対応す

るサービス属性を与える。

アクセスコントロール規則の導出は、まず、アクセスコントロール機構を持たない機器間での属性計算を行う。その後、アクセスコントロール機構を持つ機器のアクセスコントロール点の前後において、合成/継承の対象となるサービス属性が、属性評価規則に照らし合わせて合成/継承すべきか否かを判断し、それに対応したアクセスコントロール規則を生成する。

ポリシー違反の検出は、まず、その時点でのアクセスコントロール規則と属性値によってネットワーク構成記述上の全てのエンティティに関して属性計算を行う。その後、各エンティティの属性について属性評価規則と照らし合わせてみて、属性値が存在してよい条件を満たしているか否かを判断し、条件を満たしていない場合には警告を発する。

4.2 アクセスコントロールポリシー記述方法

本提案では、ポリシー記述を“オブジェクト”、“サブジェクト”、“アクション”、“制約”の組合わせで記述する。

4.2.1 オブジェクト

サービスを提供するための資源であり、“アクション”が実行される対象である。オブジェクトは対象となるシステムに依存して、指定される実体と”アクション”は変ってくるが、ネットワーク上のサービスで考えると、ネットワーク上でサービスを提供しているホストやファイルサーバ等の実体と結び付けられる。ネットワークおよびトランスポートレイヤでは、ネットワークアプリケーションサーバがサービスを行っているホストのIP アドレス・使用プロトコル (UDP/TCP)・サービス受け付けポート番号などと結び付けられる。

4.2.2 サブジェクト

ユーザなどサービスを利用することができるものを指し示す。本提案では、三つの種類のサブジェクトが指定できる。ユーザとユーザの集まりからなるグループとロールで

ある。

ユーザはネットワークシステムに接続している個人で、ポリシーのサブジェクトなり得る最小単位である。グループはユーザあるいは他のグループの集まりから構成され、階層構造を持つ。ユーザもグループも一つ以上のグループに所属できるが、グループ階層が循環することはできない。図 4.2 にグループ階層例を示す。

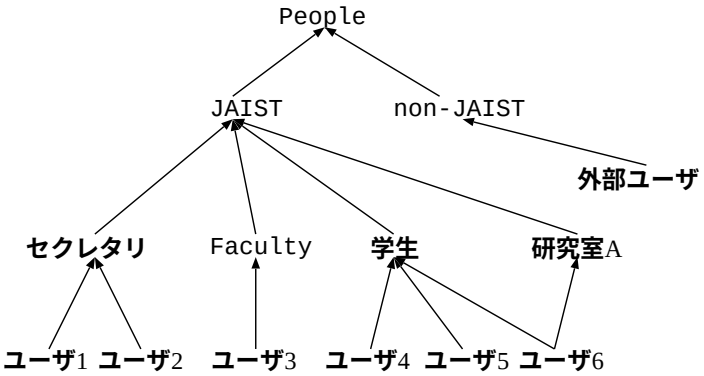

図 4.2: グループ階層例

ロールとは、特権のあつまりを指し示したもので、役割は、ユーザが行なわなければならない組織の活動を示すタスク名を示している。ユーザが何らかのタスクを行なうために、ある役割を一時的に与えられて、そのタスクを実行する事の出来るような権威を与えられる。そのため、ある役割にはそのタスクを実行することを許可される権威を持っている。グループと同様にロールも階層構造を形成する。階層の上位のロールは下位のロールを一般化した役割を示している。図 4.3 にロール階層例を示す。

ロールとグループの相違は、ロールはその時々によってユーザに与えられるものであるが、あるグループのユーザは常にそのグループメンバーであるということである。ユーザあるいはそのユーザが所属しているグループは、基本的なロールとして考えることができる。したがって、もし、ユーザがどのロールとしても活動していない場合には、ユーザはユーザ自身の権限とユーザが所属しているグループの権限をもつロールであると見ることができる。

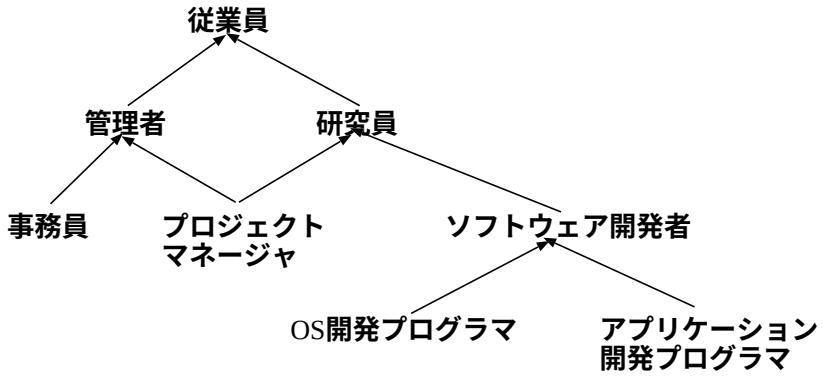

図 4.3: ロール階層例

4.2.3 アクション

アクションは、オブジェクトに対してサブジェクトが行なう操作を示している。アクションにはポジティブアクションとネガティブアクションがあり、それぞれ、オブジェクトに対して操作可能と操作不可能を表している。

4.2.4 制約

ポリシーが適用される範囲を限定するための条件を記述する。様々な条件を記述することが可能であるが、主に属性計算で利用されるのは、各種属性によって記述できる事柄と各種属性の組み合わせによって記述された制約である。

4.2.5 表記法

ポリシー記述の表記方法として [2] で述べられている ASL(The Authorization Specification Language) の表記法の一部を用いる。ASL は、論理言語であり、述語の組合わせによって、各種セキュリティポリシーを表現し、最終的にあるサブジェクトがあるロールのときにあるオブジェクトに対してアクセス可能かどうかをの判定が、ルールの数に対して線形時間で行なえるプログラム言語としての側面を持つが、本提案においては、その部分の機能は用いず、サービス記述を表現する際の表記法として、ASL の述語の一部を利用する。

今回利用した表記法を以下に示す。

cando : cando(o, s, +a)

引数に、オブジェクト、サブジェクト、アクションをとり、管理者がサブジェクトにあるオブジェクトにアクションを行なう権限を与えることを示す。

active : active(u, R)

ユーザとロールを引数に取り、ユーザがあるロールを帯びることを示す。

in : in(s_1, s_2)

2つのサブジェクトを引数に取り、あるサブジェクトがもう一つのサブジェクトのメンバであることを示す。

typeof : typeof(o, T)

引数として、オブジェクトと型をとり、あるオブジェクトが指定された型であることを示す。

アクションには許可と拒否の2つのうちどちらかで、アクション "A" に対して

$$SA = \{+a, -a | a \in A\}$$

と定義される。

例として、JAIST内部のユーザにのみサービスを行うWWWサーバのサービス記述は、次のようになる。

$$cando(\text{“}InternalWWW'', u, +access) \leftarrow in(u, \text{“}JAIST'')$$
$$cando(\text{“}InternalWWW'', u, -access) \leftarrow \neg in(u, \text{“}JAIST'')$$

4.3 サービス属性

サービス属性とは、サービス記述によって記述されたサービスに関連していることを示す属性である。あるサービス属性のネットワーク構成記述上での広がりを知ることでそのサービスがネットワーク上で影響する範囲と関係するネットワーク上のエンティティを知ることができる。

4.3.1 サービス記述から生成するサービス属性

ポリシー記述から、以下の三つの属性ラベルを生成する。

SOB : オブジェクトに対する属性

SPS : ポジティブサブジェクトに対する属性

SNS : ネガティブサブジェクトに対する属性

SPS 属性は Positive authorization policy に記述されているサブジェクトに対して生成され、SNS 属性は Negative authorization policy に記述されているサブジェクトに対して生成される。

この三種類の属性のネットワーク構成記述上の広がりにより、

- そのサービスが提供される範囲
- そのサービスを利用しようとする範囲
- そのサービスが利用できてはいけない範囲

を知ることができる。

4.3.2 属性の構成

サービス記述から生成された属性はネットワーク構成情報のレイヤに合わせて、各レイヤ毎に別のラベルを持った構成とする。また、各レイヤ毎に付属情報を持ち、それらの情報は属性計算の際に属性値と同様に合成／継承される。表 4.1 に、各レイヤ毎の属性ラベルと付加情報として何を保持するかを示す。

サービス記述より、最大 15 個のサービス属性のラベルが生成される。これらのラベルは属性計算および属性評価規則の中で用いられる。

4.4 属性評価規則

属性評価規則とは、ある属性が満たすべき条件を記述したもので、ポリシーを翻訳したものと見ることができる。サービス記述の制約をそのまま記述したり、サービス属性

レイヤ	ラベル	付属情報
アプリケーション	ap	プロトコル、ユーザ、ロール、その他
トランスポート (TCP/UDP:ポート)	tp	プロトコル、ポート
ネットワーク (IP:アドレス)	nt	IP アドレス (ソース/ディスティネーション)
データリンク (VLAN: MAC+tag)	dl	MAC アドレス+tag 情報
物理ネットワーク	ph	

表 4.1: 属性の構成

生成時に自動的に生成されるが、管理者が直接記述してもよい。

ポリシー違反検出においては、違反検出の基準として使用され、各ネットワーク上のエンティティが持っている属性が属性評価規則に違反していれば、ポリシー違反となる。

また、アクセスコントロール規則導出時には、アクセス制御点の前後のサービス属性を継承/合成してよいかの判断をする基準として使用され、合成/継承してよい場合には、そのサービスを通過させるアクセスコントロール規則を導出し、合成/継承してはいけない場合には、そのサービスを封鎖するアクセスコントロール規則を導出する。

属性評価規則は、各レイヤ毎のラベルのついた属性に対して記述を行ない、以下のような要素を用いて式で表現する。

- 簡単な制御構文 – “if 〜 else”, “else if”
- ブロック – “{”, ”}”
- 演算子 – “==”, “!=”, “||”, “&&”, “(“, “)“ “has”

また、

- アクセス強制点前後の属性 – src.attribute, dst.attribute
- ネットワーク上のエンティティの各種属性

を変数とし、定数としては属性が取り得る定数およびサービス属性そのものを使用する。

例として、ネガティブサブジェクトのサービス属性がオブジェクトおよびポジティブサブジェクトのサービス属性と同居しないようにする属性評価規則は以下のようになる。

if (*dst.attribute* **has** SOB **&&** *src.attribute* **has** SPS) {
  allow COPY SPS to **dst.attribute**;
}

if (*src.attribute* **has** SNS **&&** (*dst.attribute* **has** SPS ‖ *dst.attribute* **has** SOB))
{
  deny COPY SNS to **dst.attribute**;
}

4.4.1 サービス記述から生成される属性評価規則

サービス記述において 3 種類のサービス属性を生成したことから、以下の 2 つの条件がサービス属性に対して満たされなければならない。

- ポジティブサブジェクトサービス属性の連続性の保証
- ネガティブサブジェクトサービス属性の孤立性の保証

これは、それぞれのサービス属性がネットワーク構成記述上で以下の様な分布になっていなければならないと言うことである。

> 連続したポジティブサブジェクトサービス属性は少なくとも一点以上の個所でオブジェクトサービス属性と連続していなければならない。つまり、ポジティブサブジェクトサービス属性のみで孤立していてはいけない。
>
> ネガティブサブジェクトサービス属性は、ポジティブサブジェクトサービス属性とオブジェクトサービス属性と非接触でなければならない。

このために、属性評価規則には、次の 2 つの式が必須である。

- アクセス強制点での継承/合成を行う場合に、ポジティブサブジェクトサービス属性とオブジェクトサービス属性が存在する場合には、それぞれのサービス属性の継承/合成を許可する
- ポジティブサブジェクトサービス属性およびオブジェクトサービス属性が存在する場合にはネガティブサブジェクトサービス属性の継承/合成を許可しない (共存してはいけない)

ただし、この規則はサービス記述の対象のサービスがアクセス制御を行うレイヤで生成しなければならない。

4.5 サービス属性の初期値

オブジェクトサービス属性はオブジェクトに対応するホストに初期値として与えられる。サブジェクトサービス属性はサブジェクト (ユーザ/グループ) とネットワーク機器の対応表から対応するネットワーク機器に初期値として与えられる。

データベースなどから静的にオブジェクト/サブジェクトとネットワークエンティティとの対応が決定できる場合には問題ないが、ノートパソコンなど移動するもので対応するエンティティが決定できない場合には、対応する可能性のある全てのエンティティに初期値を与える。

4.6 ネットワーク構成記述方法

ネットワーク構成記述は、ネットワーク上の機器が接続されている状態をモデル化したものである。ネットワーク構成記述では各エンティティは、図 4.4 に示すように機器とポートから構成され機器間の接続はそれぞれのポート間がリンクによって接続されることで表される。

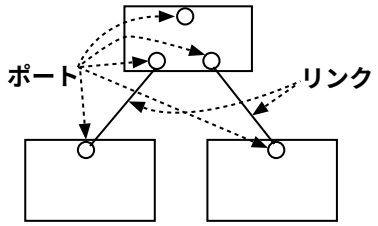

図 4.4: 機器間のポートによる接続

また、各エンティティはそのタイプに応じていくつかのレイヤから構成される。これは、アクセスコントロールはそれぞれのレイヤ毎に存在し、それら全体でサービスへのアクセスが制御されているのを反映するためである。

4.6.1 レイヤ分割とレイヤ間の接続

レイヤは以下の５つに分割される。

- アプリケーションレイヤ
- トランスポートレイヤ
- ネットワークレイヤ
- データリンクレイヤ
- 物理レイヤ

ネットワーク構成記述上の全てのエンティティにおいて全レイヤを持つわけではない。例えば、サービスを提供するサーバプログラムが動作するホストにおいては、全レイヤを持つようにモデル化されるが、ハブは物理レイヤとデータリンクレイヤしか持たない。

各レイヤ間の関係は、アクセス制御点で全レイヤの属性の関係を式で表現することで表現される。

アクセス制御点は、属性評価規則により属性計算を行なうポイントであるとともに、ポリシーを強制することができるポイントでもある。

アクセス制御点は各レイヤ毎に以下のようになる。

データリンク〜トランスポート – アクセス制御機構を持つ機器のポート間

アプリケーション – トランスポート層からアプリケーション層へのインターフェイス部分

ルータにおけるアクセス制御機構は図 4.5 に示すようになる。ルータはルータ自身がホストとして機能する部分とパケット転送機能の部分に分割されてモデル化されるため、それぞれの機能部分が持つポートと実際の入出力インターフェイス部分のポートの間の 4 個所のポイントがアクセス制御点となる。属性計算が行われる際には、各アクセス制御点の前後のポート間で属性が属性評価規則を満たすかどうかで継承/合成が決定される。この時複数レイヤを持つ機器では、それら相互のレイヤ間の関係を属性評価規則として記述することで、レイヤ間の相互作用を実現する。

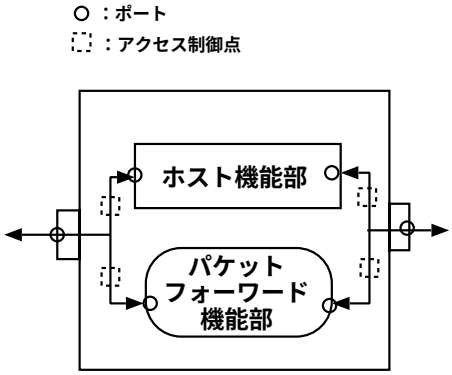

図 4.5: ルータにおけるアクセス制御点

4.6.2 各レイヤの属性

サービス属性以外にネットワーク構成記述では、各種情報を保持するために、各レイヤ毎にそれぞれ属性を持つ。

例えば、物理レイヤでは、物理的に安全かどうかや、管理区域内であるかどうかなど、機器が設置されている場所についての属性や、どこの組織の所有物であるのかを示す属性を持つ。データリンクレイヤでは、ネットワークセグメントしてどこに接続しているかを示す属性を持つことによって、ポートがバリアセグメントであるのか、内部ネットワークに接続しているのかなどの情報を知ることができるようになっている。これらの属性は必要に応じて新たに設けることができる。

4.7 属性計算によるポリシーとネットワーク構成のマッピング

4.7.1 属性計算

属性計算は、ネットワーク構成記述を構文木に見立てて行われる。リンクで単純に接続されたポート間ではサービス属性はそのまま継承属性として伝搬する。また、アクセス制御機構を持たない単純なネットワーク機器のポート間もサービス属性は継承あるいは合成される。したがって、アクセス制御機構を持ったネットワークエンティティを含

まないネットワーク部分木では、全てサービス属性が全てのポートにコピーされて同一なサービス属性を持つようになる。

アクセス制御点での計算

アクセス制御点の前後で、属性を継承/合成して良いかどうかを判定することで行う。継承/合成して良いかどうかの判断は、アクセス制御規則による。

2ポート間での判定は、以下のように行う。

1. 相手ポートの属性の差分をとる。
2. 属性一つ一つについて相手ポートの条件においてアクセス制御規則と照会して検査を行う。
3. 全てのアクセス制御規則を検査してパターンと合致しなかった場合には、デフォルト規則を適用する。

属性がアクセス制御点に到達してアクセス制御規則に基づいて計算を行なう際には、そのアクセス制御点でアクセス制御機能が実装されているレイヤに対して、対応するレイヤのラベルの属性を検査する。実装されている全レイヤで継承/合成しても良いという結果が出た場合のみ、属性の伝搬が行なわれる。(図 4.6)

伝搬される場合に、MAC アドレスなどの付属情報の変更が行なわれる。また、同じ属性が継承/合成される場合には、付属情報が引き継がれてリストして保持される。

デフォルト規則

アクセスコントロール規則あるいは属性評価規則によって判定できなかった場合に適用されるデフォルト規則を各アクセスコントロール機能を持つ機器は全論理ポート間毎にもつ。

デフォルト規則としては、

- 評価できなかったものは、継承/合成する。
- 評価できなかったものは、継承/合成しない。

の2種類があり、それぞれ、オープンポリシー、クローズポリシーを実現することになる。

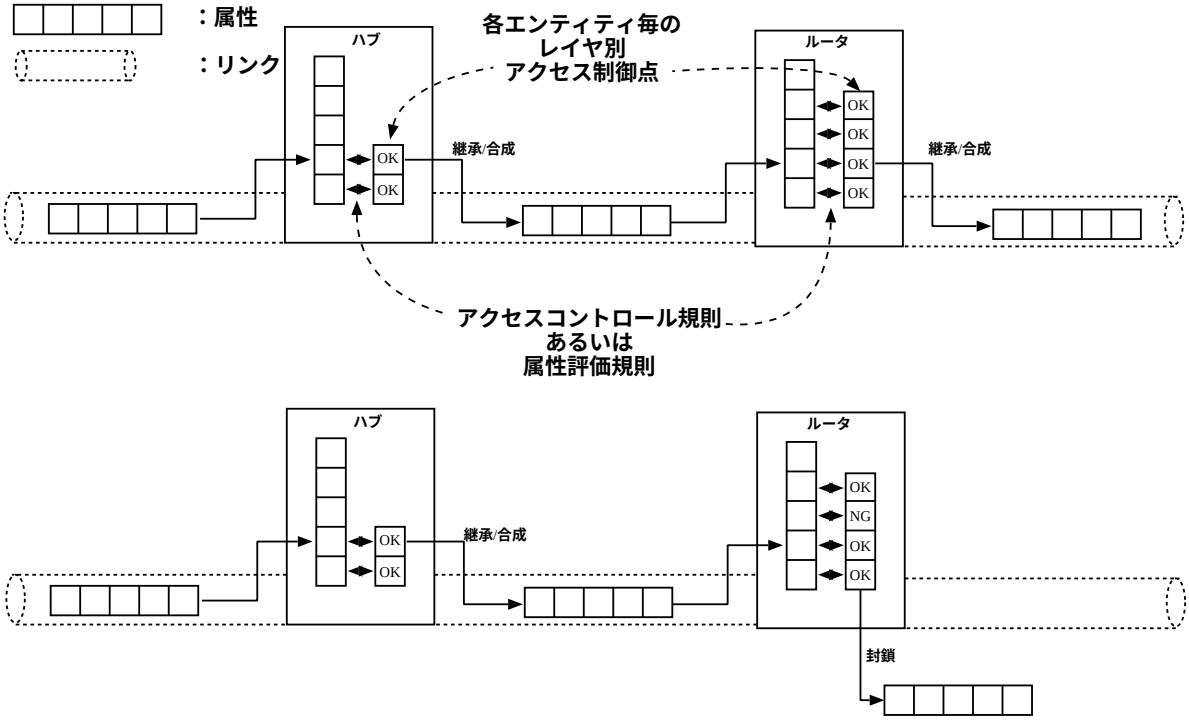
：属性
：リンク
ハブ
各エンティティ毎の
レイヤ別
アクセス制御点
ルータ
OK
OK
継承/合成
OK
OK
OK
OK
継承/合成
アクセスコントロール規則
あるいは
属性評価規則
ハブ
ルータ
OK
OK
継承/合成
OK
NG
OK
OK
封鎖

図 4.6: 属性計算の方法

4.7.2 アクセスコントロール規則の導出

アクセスコントロール規則の導出も属性計算と同様にネットワーク構成記述を構文木と見なし、リンクとポートをたどりながら以下のように計算を進める。

1. アクセスコントロール機能を持たない機器で構成されるネットワーク部分木において属性計算を行う。
2. アクセスコントロール機能を持つ機器上で全ポート間で、属性評価規則に基づきアクセスコントロール規則導出を行う。
3. 導出された規則に基づいて、全ポート間での属性計算を行う。
4. その機器に接続しているアクセスコントロール機能を持たない機器で構成されるネットワーク部分木の属性計算を行う。その際に、アクセスコントロール機能を持つ機器のポートに到達し、新たな属性でそのポートの属性を変更した場合には、その機器に対して属性が変更されたことを示す、“dirty” フラグを設定する。
5. 全てのアクセスコントロール機能を持つ機器に対して規則の導出を行った後に、“dirty” フラグが設定された機器が存在する場合には、再度、同様の導出処理を行う。
6. これを “dirty” フラグが設定された機器がなくなるまで繰り返す。

属性評価規則に基づいたアクセスコントロール規則の導出は以下のように行う。

- デフォルト規則がオープンポリシーの場合:
  - 属性計算で継承/合成してはいけない属性をポート間の関係として求める。
- デフォルト規則がクローズポリシーの場合:
  - 属性計算で継承/合成してよい属性をポート間の関係として求める。
  - 両方のポートに存在する属性についても継承/合成してよい属性として生成する。

4.7.3 ポリシー違反の検出

ポリシー違反とは、各ポートにおけるサービス属性が属性評価規則に照会して条件を満たしているかどうかである。そのため、ポリシー違反の検出を行うためには、属性計

算を行い、全ポートの属性値を求めた後、全てのポートに対して属性評価規則の条件を満たしているかを検査する。

第 5 章

プロトタイプ実装について

5.1 実装の目的と解説

今回、提案した手法を検証するためにプロトタイプシステムを構築した。本プロトタイプは、主に、ポリシーからのアクセスコントロール規則生成を行なう手法に主眼を置き、ネットワークレイヤでのみ属性計算を行なっている。

本プロトタイプで行えることは、

- **ポリシー違反検出**
- **アクセスコントロール規則の生成**

である。実フィルタリング設定を属性レベルのアクセスコントロール規則に変換する手法を実現できていないため、実フィルタリング設定の検証は行えない。実装の特徴的な部分を以下に説明する。

5.1.1 ポリシー記述

未実装であり、ポリシー記述を変換を手動で行い、ネットワーク構成記述上にマッピングを行った。また、ユーザ階層、グループ階層、ロール階層についても、特に実装は行わず、ネットワーク構成記述の設定ファイルに直接記述する方法を用いた。

5.1.2 ネットワーク構成記述

ネットワーク構成記述は、論理的なネットワーク階層のみを実装した。ネットワーク構成自身は以下の４つのデータから構成される。

pequip ：物理機器

物理的な機器を表しており、内部に論理機器を 1 台以上と物理的なポートを内蔵する。

vequip ：論理機器

物理機器に内蔵される論理的な機器を表している。論理ポートを内蔵している。また、タイプとして、以下の 3 つを持つ。

ルータ ：ネットワーク層でのルーティング機能を持ち、アクセス制御機能を持つ機器であり、アクセスコントロール規則および論理ポート間のデフォルト規則を持つ。

ハブ ：データリンク層でリピータとして機能し、アクセス制御機能は持たない。そのため、ハブの論理ポートの全ての属性は属性計算の際に無条件に継承／合成される。

ホスト ：論理ポートをひとつしか持たず、ホストとして機能する。

pport ：物理ポート

物理的なポートを表している。MAC アドレスと IP アドレスをデータとして持つ。

vport ：論理ポート

論理機器上の論理的なポートを表している。今回の実装では、属性計算を行う構文木の節として機能し、サービス属性を持つ。

例を図 5.1 に示す。この例では、物理機器と論理機器が以下のような対応になっている。

“host1”,”host2”,”host3”が同一ハブに接続されているが、論理的には、“vhost1”と”vhost2”,”vhost3”と”vhost4” の 2 つのセグメントに分離されたネットワークになっている。この例のネットワーク構成記述の設定コマンドを付録 B に添付する。

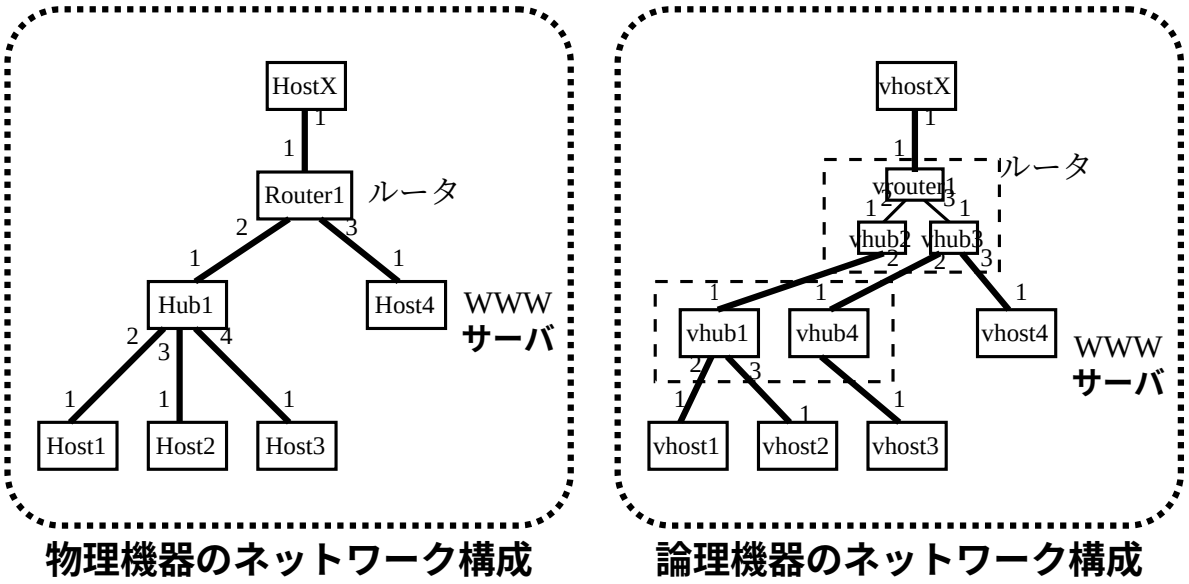

図 5.1: ネットワーク構成記述例 (物理構成と論理構成)

物理機器	論理機器
Router1	vrouter1, vhub2, vhub3
Hub1	vhub1, vhub4
Hostx	vhostx
Host1	vhost1
Host2	vhost2
Host3	vhost3

表 5.1: ネットワーク構成記述例（物理機器と論理機器の対応）

5.1.3 属性の種類

本実装では、表5.2に示す属性を設け、属性評価マトリックスで使用できるようにした。

名称	格納先	解説	継承/合成
所属属性	pequip	機器の所有	×
管理グループ属性	pequip	機器の実質的な管理グループ	×
ロケーション属性	pequip	機器の設置場所	×
接続属性	vport	どのネットワークに接続しているか	×
管理属性	vport	管理下か否か	×
サービス属性	vport	サービス属性	○
信頼度属性	vport	信頼してよいユーザが利用しているかどうか	○

表 5.2: 実装した各種属性について

ロケーション属性、接続属性および信頼度属性は、それぞれ、表 5.3、表 5.4、表 5.5 の値を取る。

値	意味
LF_SECURE	物理的な安全である
LF_UNSECURE	物理的に安全でない
LF_INAREA	管理領域内である
LF_OUTAREA	管理領域外である
LF_PRIVATE1	立入り制限領域 1
LF_PRIVATE2	立入り制限領域 2
LF_PRIVATE3	立入り制限領域 3
LF_PRIVATE4	立入り制限領域 4

表 5.3: ロケーション属性の値

値	意味
PA_CONNOTSET	属性未設定
PA_CONOUT	外部接続ポート
PA_CONIN	内部接続ポート
PA_CONDMZ1	バリア 1 接続ポート
PA_CONDMZ2	バリア 2 接続ポート
PA_CONDMZ3	バリア 3 接続ポート

表 5.4: 接続属性の値

値	意味
PA_UNKNOWN	未知
PA_ULTIMATE	最大限信用する
PA_COMPLETE	完全に信用する
PA_MARGINAL	ある程度信用する
PA_UNTRUSTED	信用しない

表 5.5: 信頼度属性の値

5.1.4 属性評価マトリックス

属性評価規則は式ではなく、あるサービス属性が存在しても良い条件あるいは存在してはいけない条件を各属性のパターンとして記述した属性評価マトリックスとして実装した。あるサービス属性について満たすべき各種属性のパターンを記述し、サービス毎にマトリックスにする。

5.1.5 属性計算方法

本実装では、ネットワーク構成記述の vport を属性計算の構文木の節と見なして、属性計算を行うようにした。

信頼度属性の計算

信頼度属性の属性計算は、以下のルールで行った。

- デフォルト規則がオープンポリシーの場合は継承/合成する。
- デフォルト規則がクローズドポリシーの場合は継承/合成しない。

信頼度属性が継承/合成される際には、属性値は必ず信頼度の低いほうになる。

循環回避方法

今回、論理ネットワークにおける計算であり、VLAN を意識している。したがって、属性計算の循環させない方法としてはコリジョンドメイン内でのスパニングツリーを計算し、スパニングツリーに沿って属性計算を行うことで計算の循環を起さないことが可能である。但し、今回の実装では計算の循環を防ぐために、論理機器に属性計算が行われたということを示すフラグを設けた。属性計算が行われた時にフラグをセットし、フラグがセットされている論理機器は再度計算しない。

5.2 例題

5.2.1 組織内部のみでアクセス可能な WWW サービスの例

図 5.2 のようなネットワーク構成において、組織内部のみでアクセス可能な WWW サービスを提供するときに必要なアクセスコントロール規則の導出を行う。

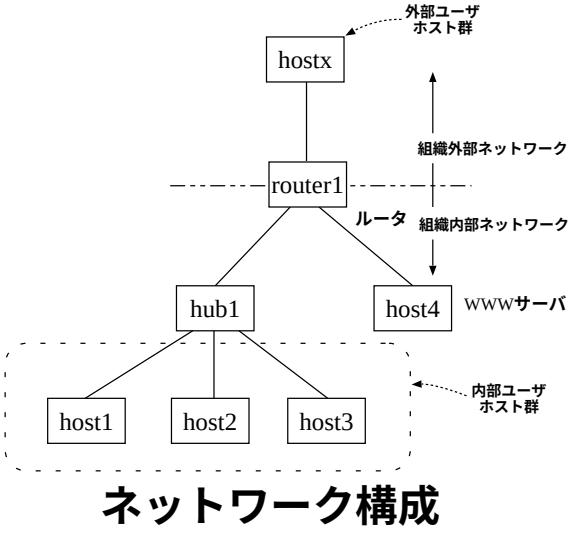

図 5.2: 例題①: ネットワーク構成

Router1 を介して組織内部と外部ネットワークが接続されており、同じルータを介して、WWW サーバの設置されているネットワークとその利用者が存在するネットワークが接続している。

この時、サービス属性として、

WWW Client+ – ポジティブサブジェクト属性

WWW Client- – ネガティブサブジェクト属性

WWW Server – オブジェクト属性

が生成される。

また、生成された属性間の関係から、表 5.6 に示すような属性評価マトリックスが生成される。

属性値	アクション	サービス属性	所属属性	管理グループ属性	ロケーション属性	接続属性	管理属性	信頼度属性
WWW Server	allow	WWW Client+	N/A	N/A	N/A	N/A	N/A	N/A
WWW Server	deny	WWW Client-	N/A	N/A	N/A	N/A	N/A	N/A
WWW Client+	allow	WWW Server	N/A	N/A	N/A	N/A	N/A	N/A
WWW Client+	deny	WWW Client-	N/A	N/A	N/A	N/A	N/A	N/A
WWW Client-	deny	WWW Server	N/A	N/A	N/A	N/A	N/A	N/A
WWW Client-	deny	WWW Client+	N/A	N/A	N/A	N/A	N/A	N/A

表 5.6: 例題①: 属性評価マトリックス

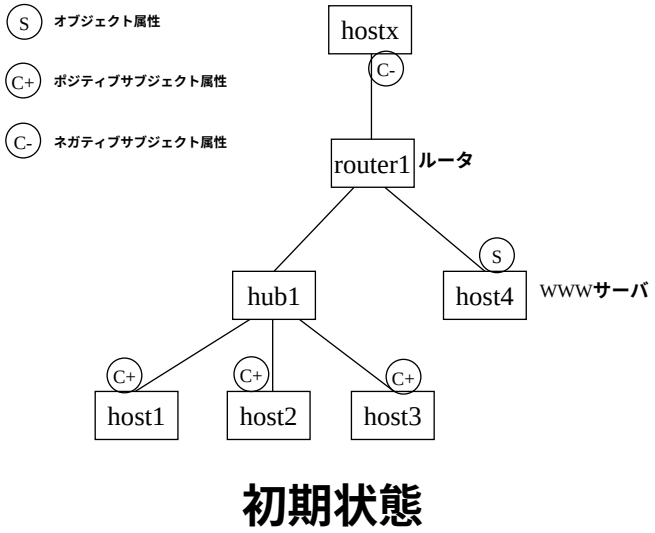

図 5.3: 例題①: 初期状態

属性値の初期状態は図 5.3 に示すようになっている。

このネットワークに対して、アクセスコントロール規則を生成する。

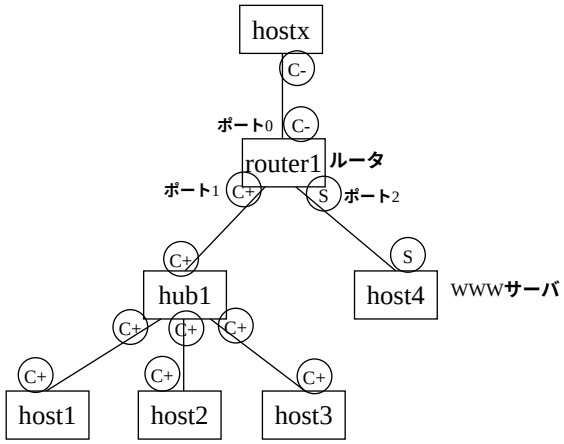

図 5.4: 例題①：部分木の属性計算

まず、アクセスコントロール機能を持たない機器で構成されるネットワーク部分木において属性計算を行う。このネットワークの場合、アクセスコントロール機能を持つのは” Router1”のみであるので、属性計算を行うことで、“Router1”の全てのポートの属性が計算できる。（図 5.4）

次に、“Router1”の全ポート間で属性評価マトリックスに基づきアクセスコントロール規則導出を行う。“Router1”のポート 0 とポート 1では、表 5.6 から WWW Client+と WWW Client-の関係はともに “deny”であるので、ポート間で両属性を伝搬しない以下のようなアクセスコントロール規則を生成する。

```
attrfilter "WWW Client-" "Router1" 0 1 block ;
attrfilter "WWW Client+" "Router1" 1 0 block ;
```

ポート 0 とポート 2 間では、WWW Server と WWW Client- の関係から以下のようなアクセスコントロール規則を生成する。

```
attrfilter "WWW Client-" "Router1" 0 2 block ;
```

```
attrfilter "WWW Server" "Router1" 2 0 block ;
```

同様にポート 1 とポート 2 の間では、WWW Server と WWW Client+の関係はともに “allow” であるので両属性がポート間を伝搬することができるアクセスコントロール規則が生成される。このとき、WWW Server 属性あるいは WWW Client+属性が双方のポートに伝搬された後に再度アクセスコントロール規則の生成が行われるため、両属性に対して双方向でのアクセスコントロール規則が生成される。

```
attrfilter "WWW Client+" "Router1" 1 2 pass ;
attrfilter "WWW Server" "Router1" 1 2 pass ;
attrfilter "WWW Client+" "Router1" 2 1 pass ;
attrfilter "WWW Server" "Router1" 2 1 pass ;
```

この例では、アクセスコントロール機能を持つのが “Router1” だけであったので、他のアクセスコントロール機能を持つ機器のアクセスコントロール規則生成による影響を受けなかったため、これでアクセスコントロール規則生成が終了する。他にアクセスコントロール機能を持つ機器がある場合には、全ての相互影響が解消されるまで、つまり、“dirty” フラグが無くなるまで、規則生成計算を行うことになる。

最終的な属性の分布は図 5.5 のようになる。

5.2.2 ポリシー違反を VLAN 設定変更により修正する例

図 5.6 に示す様な構成のネットワークにおいて、あるユーザ”U”がオフィス領域 Aからオフィス領域 Bに移動し、使用するホストが”host2”から”host3” に変更になった場合を考える。

この時、ユーザ”U”が使用するサービスが信頼度属性 “untrust” の端末では使用できないサービスで、“host4” を利用しているユーザが外部派遣者で信頼度属性が”untrust” であった場合を想定する。

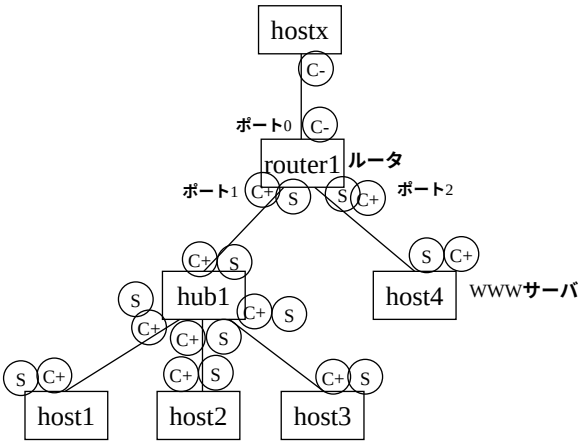

図 5.5: 例題① : 最終的な属性分布

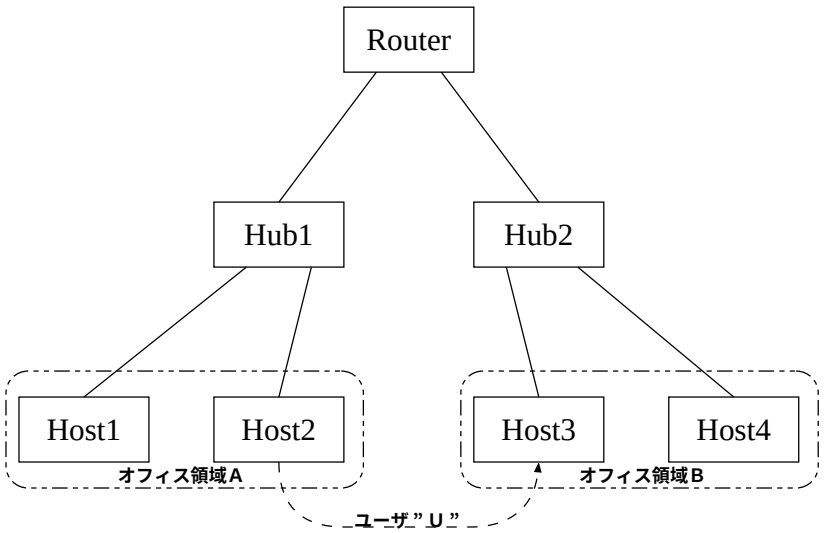

図 5.6: 例題② : ネットワーク構成

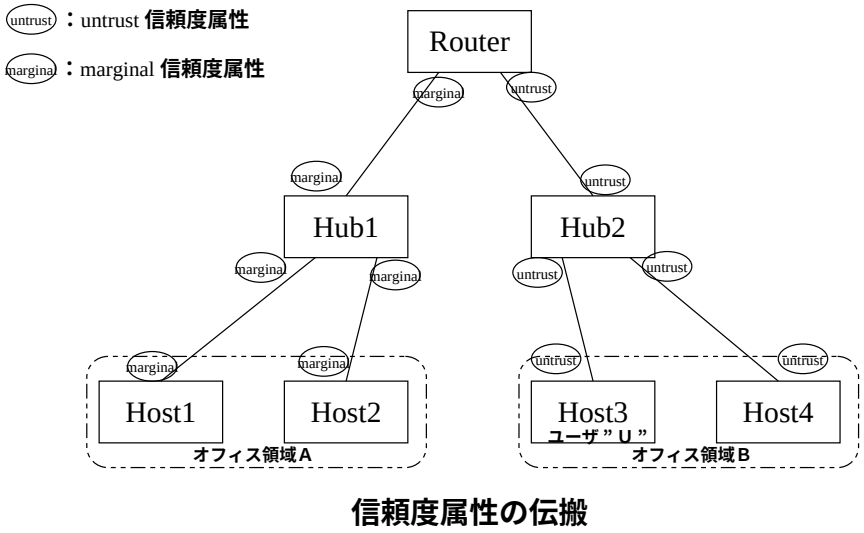

図 5.7: 例題② : 信頼度属性の伝搬

このままのネットワーク構成では、図 5.7 に示すようにユーザ”U”が、サービスを利用すると信頼度属性が “untrust” である”host3”でサービスを利用することになり、ポリシー違反となってしまう。

そこで、ネットワーク構成を図 5.8 に示すような構成に変更し、“Router”と”Hub2”において VLAN を設定し、仮想ルータである”Verouter”で適当なアクセスコントロール規則を設けることで、“Host3”を “Host4”から分離する。

こうすることで、信頼度属性は、図 5.9 に示すように、“Host3”において “marginal”となり、ユーザ”U” の移動に伴うポリシー違反を取り除くことが可能となる。

5.3 プロトタイプから得られた考察

5.3.1 計算順序による属性アクセスコントロール規則生成への影響

計算順序によって属性アクセスコントロール規則生成の結果が異なってくる場合がある。

5.10 に示すように、あるサービスのポジティブサブジェクトであるクライアントが “Host3”、ネガティブサブジェクトであるクライアントが “Host2”を利用しており、オブ

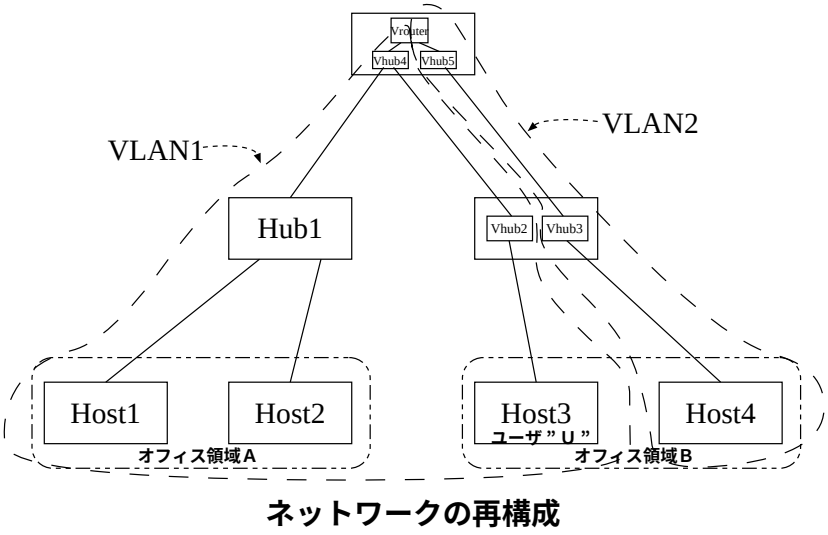

図 5.8: 例題②：ネットワークの再構成

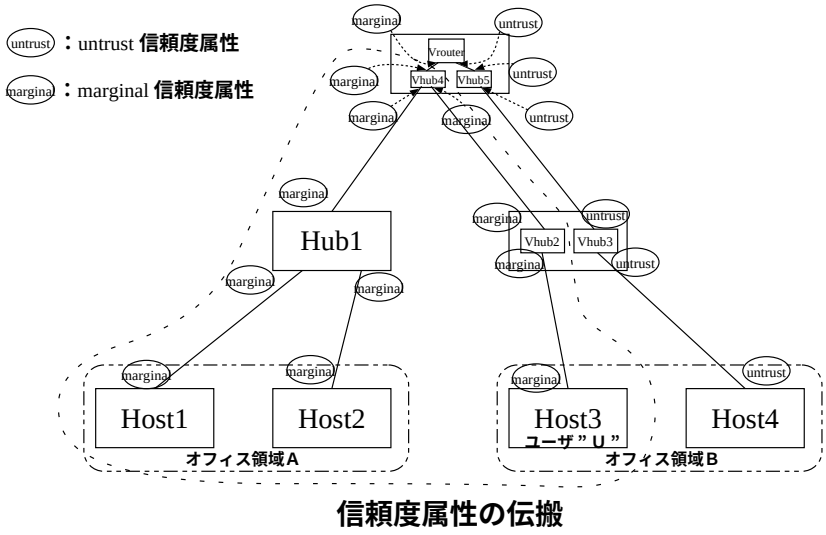

図 5.9: 例題②：信頼度属性の伝搬の分離

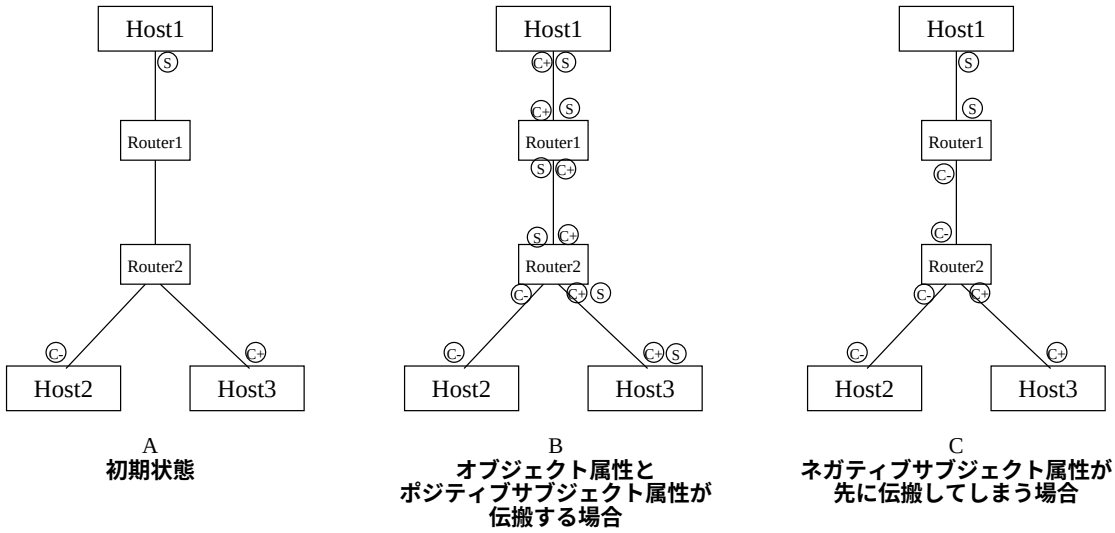

図 5.10: 属性計算順によるアクセスコントロール規則生成への影響

ジェクトであるサーバ “Host1”でサービスを行っている場合を考える。

このとき、属性評価マトリックスとして以下のものを定義する。

- ⓢ と ⓒ+ は共存しなくてはならない。
- ⓢ と ⓒ- は共存してはならない。
- ⓒ+ と ⓒ- は共存してはならない。

このとき、規則生成の計算の順番としては、

1. Router1 からスタートする場合
2. Router2 の”Host3” 側のポートからスタートする場合
3. Router2 の”Host2” 側のポートからスタートする場合

が考えられる。はじめの 2 つの場合には、Router2 で ⓒ- をブロックし、ⓒ+とⓢがうまく共存する規則が生成される。(5.10 の B) しかし、最後の場合には、ⓒ- が Router1 と Router2 の間のリンクに伝搬してしまい、所望のアクセスコントロール規則が生成できない。(5.10 の C)

この問題を解決するために 2 つの方法が考えられる。

- ネットワーク構成の回りから一つずつ属性計算を行うようにしてアクセスコントロール機構を持った論理機器に来るたびに規則生成計算を行う方法
- オブジェクトサービス属性 ⓢ とポジティブサブジェクトサービス属性ⓒ⁺の属性計算/規則生成計算を先に行い、その後ネガティブサブジェクトサービス属性の属性計算/規則生成計算を行う方法

それぞれの特徴としては、前者の場合、ネットワーク構成全体に対する知識を必要とする。また、その知識に基づいてスパンニングツリーを計算して、属性計算木のリーフから属性計算/規則生成計算を行っていく処理を行わなければならない。また、後者の場合、属性値にポジティブ／ネガティブの区別を行うための情報を付加する必要がある。ネットワーク構成全体に対する知識は必要ない。

前者の方法では、スパニングツリー計算処理が入ることと、リーフからの計算順をどのように行えばよいかの問題があり、実現が容易でないと考え、今回は後者の方式をとった。

5.3.2 実装上の問題

属性評価マトリックスの表現能力

現状の実装では属性評価マトリックスのエントリ間での関係 (or, and, group) などが表現できないため、ポジティブ/ネガティブ属性が混在するようなネットワーク構成では必ずポリシー違反になってしまう。(図 5.11)

これは、属性のラベルをレイヤ毎に個別に持たせるようにし、それらの属性ラベルを用いて属性評価規則を記述できるようにすることで解決できる。

論理ネットワークレイヤのみ

アプリケーション層でアクセス制御を行うことが適したサービス記述を表現することができない。例えば、『ユーザが事務員か faculty であれば、事務用 WWW サーバにアクセスできるサービス』を考えると、faculty と学生は混在して同一セグメント上に存在する事を考えると、論理ネットワークレイヤでの属性計算では、ポジティブサブジェ

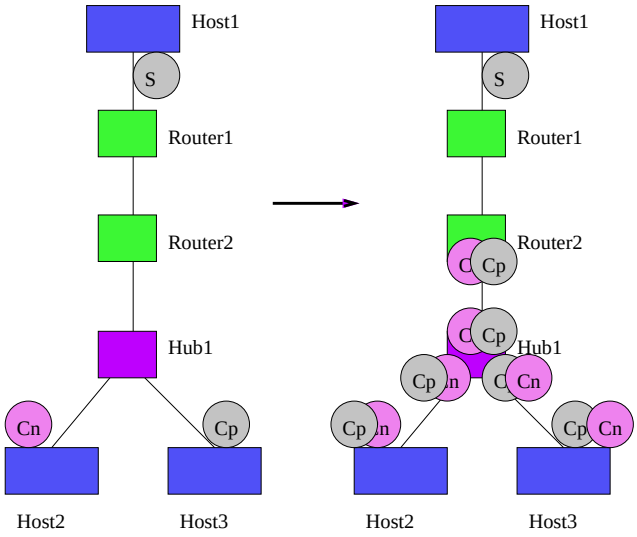

図 5.11: ポジティブ/ネガティブ属性の混在

クト属性とネガティブサブジェクト属性が共存することになり、ポリシー違反になってしまう。サービスがアプリケーションレイヤにおいてアクセス制御されるのが適しているような場合には、アプリケーションレイヤの属性のみで属性評価規則を記述すれば、ネットワークレイヤでのポリシー違反を起すことはなく、適切な評価を行なうことができる。

5.3.3 ネガティブサブジェクトの記述

サービス記述においてネガティブサブジェクトを記述しなかった場合、扱い方によって問題が生じる。

1. ポジティブサブジェクトの以外のすべてのオブジェクトをネガティブオブジェクトとして扱う方法を用いた場合、ポリシー記述者が意図的にネガティブサブジェクトを記述しなかったとしたら、記述者の意図と不整合を起こしてしまう。
   例えば、接続属性により内部にのみ接続している条件を指定することで、直接ネガティブサブジェクトを指定せず、アクセスコントロールしようと意図していた場合などが考えられる。
2. そのままネガティブサブジェクト無しで属性計算を行う場合、ポリシー記述者が別

の評価規則を記述していなかった場合あるいは、記述し忘れの場合に、アクセスを制限するアクセスコントロール規則をうまく生成できず、広範囲にサービスを提供することになる。

どういう意図で記述したのかを示す方法か、ポリシー記述者が特に必要ない場合には、ネガティブサブジェクトの記述を省略できるような記述方法が必要である。

FAM[3]では、FAM Authorization language に対してより容易にポリシー記述を行えることを目的に、Compact Authorization Manager (CAM) とその記述言語 を用意している。

5.3.4 ロールのネットワーク構成記述上へのマッピング方法

現状では、ロールをうまくネットワーク構成記述上に表現することはできない。そのための方法として新たにロール限定子付きサブジェクト用サービス属性ラベルを考えアプリケーションレイヤのネットワーク構成記述において属性評価を行うことで、ロールを扱うことができると考える。

5.3.5 アプリケーション層でのポリシー違反の影響範囲の特定

ネットワーク構成記述では、レイヤ構造になっているために、ネットワークレイヤにおいて安全が保たれていたとしても、上位のレイヤにおいて信頼度属性が安全でなくなった場合には、それに影響されて、ネットワークレイヤにおいても信頼度属性が低くなるはずである。例えば、アプリケーション層で root 権限が不正取得された場合などである。その影響範囲は以下のように計算される

- まず同一論理機器の属性に影響は及ぶ。そのため、同一論理機器上の全ての属性について影響範囲を測定する必要がある。
- 論理機器の属性は論理ネットワークを通じて伝搬しており、関係している全ての属性に対して連続している範囲を求めることで、影響範囲が特定する。
- 論理機器の属性とアプリケーション層の属性の関係を定義する式を与えることで

その範囲内の関係する属性の値がどのように影響するかを定義する。

- 先ほど求めた影響範囲において、影響を受ける論理機器属性の値を与えた式によって求める。

このように計算した後、ネットワーク構成記述上の全てのエンティティに対して、ポリシー違反検出を行い影響範囲と影響度を求める。

5.3.6 サービス定義の粒度

今回の実装では、サービスは単一オブジェクトのサービスを仮定していたが、複数オブジェクトから構成される複合サービスを記述することも考えられる。また、サービス定義自身をオブジェクトすることで、サービスのグループ化したサービス定義を行うことも可能と考える。サービス記述のバリエーションとしてどのようのものがあるかについては、さらに見極める必要がある。

5.3.7 デフォルト規則の粒度

今回の実装では、デフォルト規則は論理ポート間毎に設けたが、デフォルト規則を設定する点としては、以下の 3 種類が考えられる。

- 論理機器毎
- 論理ポート間毎
- オブジェクト単位/サービス単位

また、これらの組み合わせて、オブジェクト単位の規則からはじめて論理機器全体のデフォルト規則に至るまで、規則が設定されているかどうかを探索することも考えられる。実装としては、デフォルト規則を自由に設定できるようにしておき、実際の機器の設定がどのようになっているかによってデフォルト規則の持ち方を決定すべきである。

第 6 章

考察

本研究を通して遭遇した問題点について、ポリシーベースのシステム管理手法における課題として考察した点を述べる。

6.1 ポリシー

本論文で扱ったポリシーは、セキュリティポリシー、特にアクセスコントロールに関するポリシーで、各種ポリシーの一部であった。組織がネットワークを運用していく上でのポリシーにはそれ以外に、マネージメントポリシーやポリシーのためのポリシーなど様々な種類がある。組織のネットワークをポリシーに基づいて一元管理することを考えると、これら様々なポリシーも何らかの枠組みで扱える必要がある。全てのポリシーがネットワーク上に展開されて何らかの属性計算を行なうことで扱えるわけではないのは明らかであり、ポリシーをデータとしてそのまま扱うような手法を考える必要がある。

ポリシーの表現手法としても、今回はオブジェクト的な手法を取ったが、自動検証が必要な場面では仕様記述言語によるポリシーの表現を用いる方が良い場合もあろう。フォーマルメソッドだけではコンピュータセキュリティポリシーを扱いきれないので、ヒューリスティックな理由付けと組み合わせる [7] ことも提案されており、様々なポリシーを扱う上で表現手法も工夫する必要ある。ポリシー記述を宣言的記述するか、状態機械として記述するかも議論のあるところである。状態機械で記述すると個々の設定やポリシーがネストする状態で記述することになり、宣言的記述に比較して、ポリシーコンフリクトの検出が行ないにくい。

また、サービス記述の制約の記述において利用可能時間の制約など時間的な事象に対する条件を記述することができるが、属性によって表現することが難しく今回のシステムでは属性計算することができない。そのため、サービス記述に時間的な制約は記述しても属性計算に反映できない。

IETF Policy-framework ワーキンググループでは、ポリシーの時間的な制約について、ほとんどのデバイスが時間あるいは日時に基づいた設定を行う機能がない現状で、以下の方法で取り扱う可能性があると述べている [5]。

- ポリシー管理ツールがポリシーリポジトリーのポリシールール格納を時間毎に行う。
- ポリシーコンシューマが時間的な記述を解釈し、時間毎にポリシーターゲットの設定を変更する。

前者では、リポジトリーを動的に変更することになり、現状想定されているリポジトリーデータが静的であることと整合が取れなくなる。また、ポリシーコンシューマがポリシーを一時保持している場合に、リポジトリーの変更時間と実際にポリシーがポリシーターゲットに適用されるまでの時間的ずれが生じることが考えられる。後者では、ポリシーコンシューマに依存するが、時間的ポリシーが実際に有効になるまでポリシーコンフリクトについて検出できない場合が考えられる。ポリシーコンフリクトは複数のポリシー間での不整合であるから、時間によってポリシーが変更される場合に全ての時間に渡って、全ての場合を尽くした形での事前のコンフリクト検出を行うことが現実的でない場合である。ポリシーの時間的制約に対する記述の処理方法も検討する必要がある。

ポリシーエディタの機能として述べたが、抽象度の高い表現による組織のポリシーを管理してうまく運用することが非常に重要である。そのために、

- 各種ポリシー間の関連の維持管理
- 個々のポリシーの責任者のトラッキング
- ポリシーと罰則規定などの関連
- ポリシーの有効期限管理
- ユーザのポリシー受諾のサインの管理

などの機能を持ちポリシーに関連する様々なタスクを一元管理して、組織のポリシーを有効活用できるようにするツールが今後必要である。

6.2 アクセスコントロール

今回、アクセスコントロールに関しては、サービス属性レベルでのアクセスコントロール規則を導出し、実際の機種のフィルタリング記述に変換するところまで行なわなかった。サービス属性に付属情報としてフィルタリング記述に必要な情報は網羅されているので、機種依存のフィルタリング記述に変換することは可能である。しかし、ベンダーや機種に依存したフィルタリング記述は様々なものがあるので、それらに直接変換するのではなく、メタフィルタリング言語を考案し、一旦、その形式に変換したものを各種フィルタリング記述に変換することを考えるべきであろう。

メタフィルタリング言語として、今回、調査したものでは、FLC(Filtering Language Compiler)[1]があり、一つのメタフィルタリング言語から

- Cisco (extended access-lists)
- IP Filter
- ipfw
- ipfwadm
- ipfirewall
- screend

などのフィルタリング記述を生成できるものである。しかし、1995年から開発は止っており、最新版のフィルタリング記述に追い付いていない。また、これらのフィルタリングはステートレスなパケットフィルタリングであり、ステートフルなフィルタリングや、最近、出てきたレイヤ4以上でのフィルタリングを行なうようなものへの対応を考えると、現状では適当なメタフィルタリング言語はなく、今後、研究する必要がある。

6.3 ポリシーコンフリクト

今回の提案では、ポリシーコンフリクトは、以下の地点でおこると考えられる。

- サービス記述間のコンフリクト
- 一つのサービス記述の属性評価規則内でのコンフリクト
- 複数のサービス記述から生成された属性評価規則間のコンフリクト
- アクセスコントロール規則内のコンフリクト

サービス記述間のコンフリクトの検出は、複数のサービス記述でオブジェクトの重なりがあるかどうかで判断することができる。もし、重なりが無いなら、コンフリクト無いと考えられる。重なりがあるばあいには、コンフリクトの可能性があると考えサービス記述の内容を検査する。コンフリクトには、

- 様相の矛盾 – 一方のサービス記述で操作を許可し、別のサービス記述では、操作を禁止していた場合
- 動機と権限の不整合 – 権限がないにも関わらず、何らかの操作をしようとしていた場合

が考えられる。前者の場合には、両方のサービス記述が必要であるならば、優先順位を与えることで解決する。後者の場合には、ポリシー記述が不十分であるので、管理者が矛盾を無くするように調整するしかコンフリクトを解消する方法はない。

一つのサービス記述の属性評価規則内でのコンフリクトの検出は、個々の属性評価規則の式をを自分自身の属性評価規則と照会して、結果が矛盾するかどうかで行う。コンフリクトを修正するための優先順位は、属性評価式の順番であらわされる。

複数のサービス記述から生成された属性評価規則間のコンフリクトについても同様に、属性評価規則を属性評価規則で照会することで、検出できる。

アクセスコントロール規則内のコンフリクトの検出は、一つのサービス属性に対して矛盾した規則が存在しないかどうかをチェックすることで行う。

IETF Policy-framework のフレームワークでは、ポリシーコンフリクトに関しては、２つの機構があると述べている [17]。一つはグローバルコンフリクト検出で、リポジト

リーにポリシールールを格納する前にポリシー管理ツール側で行われる。特定のデバイスやネットワークに依存していない部分でのコンフリクト検出である。ポリシールールの静的な検査であり、時間制約や動的に変化する情報に基づいたポリシールールのコンフリクトは検出できない。

もう一つが、ローカルコンフリクト検出で、ポリシーコンシューマの機能として該当するポリシーコンシューマが制御している全てのポリシーターゲット間のコンフリクトを検査する。ローカルコンフリクト検出で行われる検査は

Conflict Detection – 新しく生成されたり、変更されたり、削除されたポリシールールと既存のルールとの整合性の検査

Requirements Checking – ポリシーが必要としている各種資源が手当て可能かどうかの検査

Feasibility – 全てのポリシーが使おうとしているサービスに見合うだけのサービスをネットワーク全体として提供できるかの検査

がある。

しかし、ポリシーコンフリクト自身のアルゴリズムについては、実装上様々な方法が考えられるとして、言及をしておらず、情報モデル [14] とデータスキーマ [18] において、ポリシールールの優先度を与える属性を定義しているのみである。

6.4 ネットワーク構成モデル化

今回の提案では、ネットワークを５つのレイヤとしてモデル化した。属性計算する場合に、アプリケーションレイヤは下位レイヤ間の接続を介して計算が伝搬し、伝搬するネットワーク構成のパスはアプリケーションレイヤの情報によって変更されないと仮定している。しかしながら、最近、製品化されつつある URL によってフィルタリングを行う機器やトランスペアレント Web キャッシュなどは、アプリケーションレイヤの情報である URL の値によってサービス属性が伝搬するネットワーク構成上のパスが変更される可能性がある。ネットワーク構成のレイヤを増やすか、サービス記述粒度で対応する方法が考えられるが、今後の課題である。

また、IPSec、IP in IP や MPLS(MultiPotocol Lavel Switching) などの技術により、ネットワークトンネルを構成することが出来る。今回、これらのネットワーク構成記述については考えなかったが、今後記述できるようにしなければならない。

6.5 ポリシー違反の自動除去

ポリシー違反を自動除去するには、ポリシー違反除去する方法のを発見し、それらを定量的に評価するための指標を確立する必要がある。ポリシー違反を除去する方法を見つけ出す手法としては、どのようなポリシー違反においてどのような処理を行える可能性があるのかを事例データベースとして蓄積し、その中から該当するポリシー違反に近い事例を検索し、ヒットした事例が取った方法を列挙する方法が考えられる。つぎに、これらの事例から取り出した解決手段を定量的に評価する必要があるが、ある程度の評価基準を設けることは可能であるが、実際には、その組織のポリシーあるいは、状況によってどの解決手段がもっとも良いものであるかの判断をする必要があるため、固定的な評価基準を設けることは不可能である。例えば、不正なパケットをネットワーク上に流している機器があって、それらが他組織に影響を与えないようにする必要がある場合に、2 つの解決方法

1. それらの機器をリプレースする方法
2. 外部に出るところで、フィルタリングする方法

解決としては、前者が根本的な解決であるが、費用の面やそれらの機器が複数の内部組織に跨って調整が難しい場合などで、後者の解決策を取らざるを得ない状況などは、定量的な指標で持って扱うことはできない。

したがって、ポリシー違反の自動除去を行うよりも管理者がポリシー違反を解決するためのサポート機能を提供することを主に考えるべきであろう。

6.6 属性計算の QoS 制御への応用の可能性

QoS をスペクトラムを持ったアクセス制御と考え、属性計算によって QoS に対するポリシーの要求がネットワークのサービス提供能力に対して妥当かどうかを検査できる。必要なのは、リンクをネットワーク構成記述のネットワークエンティティとしてモデル化と、各 QoS を要求しているサービス毎に必要なリソースとリンクが提供可能なサービス能力との関係式をリンクの正確に応じて設けるような属性計算モデルである。

6.7 組織におけるセキュリティ情報の一元管理

組織の情報一元管理をしっかりしたものにするためには、セキュリティ管理の最小基準単位は個人で行えなくてはならない。全ての個人個人に対するアクセスコントロールリストをメンテナンスすることは現実的ではない。そこで、個人個人が組織内部のリソースに対するアクセス可能な度合いを示すセキュリティレベルの属性を持たせる。また、組織内の全てのリソースに対してそのリソースが守られるべき度合いを示すプロテクションレベルを設ける。リソースに対するアクセスコントロールは、通常は、アクセスしてきたユーザのセキュリティレベルとリソースのプロテクションレベルとの関係から決定される。

ただし、ある役職であるとか特定の個人に対して、あるリソースに対しての特別なアクセス特権を提供したり、逆にアクセスを拒否したりすることができるように、リソース毎にアクセスコントロールを持ち、通常のアクセスコントロール制御をオーバーライド可能とする。また、このアクセスコントロールリストを個人毎に持てるようにすることも考えられる。

個人の属性を一括管理する必要があるが、会社など一般の組織では、通常行われていることであるし、問題にはならないと考えられる。難しいのは、アクセスしてきた個人を特定することと、個人の属性を問い合わせるタイミングをいつにするかであろう。

そのために、個人認証の枠組みを整備する必要があるのと、アクセスをトリガーとして、ポリシーリポジトリーおよび個人属性データベースへの問い合わせを行うリファレンスモニターを実現することが必要である。また、アクセスしたものを特定するための手段あるいは、プロトコルを開発する必要がある。

6.8 小型デバイスによるリファレンスモニターの実現

汎用的なリファレンスモニターを実現することは一般には非常に困難であるが、全てのネットワークデバイスのNIC部分に次のような機能を持った小型デバイスを設置できるとすると、ネットワークアクセスに対してのリファレンスモニターを実現できる。

- 設置されているNICへの全てのアクセスをインスペクトする
- ポリシーの指示にしたがってアクセス制御を行なう
- ユーザとIPアドレスのマッピングが行なえる場合には、IPアドレスからユーザを特定してアクセス制御を行なう

ネットワークの規模が大きくなると、全てのネットワーク機器に小型デバイスを設置することが難しくなり、現実的ではなくなる。しかし、ある程度の規模で、セキュリティが非常に重要なある程度閉じたネットワークにおいては、非常に有効な統合的ネットワークセキュリティ管理手法と考えられる。例えば、病院や地域医療のネットワークなどである。

また、小型デバイス同士で必ずIPSecなどのセキュアな通信路を用いるようにすれば、全てのセッションにおいてエンドツーエンドのダイナミックなVPN環境を構築できる。

6.9 ユーザとIPアドレスの一対一のマッピング手法

いろいろな手法が考えられる。まず、ダイナミックDNSとDHCPの組み合わせと、ユーザがログインしたときのホスト名の情報からマッピングを行なう方法がある。また、別の方法として、アクセスがあったときに、アクセスしてきたソースホストに対して、問い合わせのメッセージを送り、個人認証用証明書を獲得して接続してきた個人を特定する方法が考えられる。

ユーザとIPアドレスの一対一のマッピングが行なえれば、全てのレイヤのアクセス制御をIPアドレスによるアクセス制御に変換することが可能である。

第 7 章

おわりに

7.1 まとめ

本論文では、組織におけるネットワークにおいてセキュリティの各種設定の整合性を保った状態で運用管理を行うためのアプローチとして、ポリシーに基づいたネットワークの一元管理システムの枠組みを提案した。さらに、ネットワークを提供されるサービスの集合であると考え、サービスをポリシーによって記述したサービス記述によってネットワークを定義し、サービス記述のうちアクセスコントロールポリシーから 3 つのサービス属性

- オブジェクトサービス属性
- ポジティブサブジェクトサービス属性
- ネガティブサブジェクトサービス属性

を生成し実際のネットワークをモデル化したネットワーク構成記述上で属性計算を行うことで、サービスがネットワーク上でどのような広がりで展開されているかと、関係しているネットワークエンティティを知る手法を提案した。このネットワーク構成記述上のサービス属性の属性計算手法により、アクセスコントロール機能を持つ機器において生成されるべきアクセスコントロール規則の導出とポリシーに違反しているネットワークエンティティを検出する手法を提案し、プロトタイプを構築して有効性を確認した。

また、ポリシーに基づいたネットワークセキュリティ管理システムにおける問題点についても議論し、今後の展望を考察した。

7.2 今後の課題

今回は、属性計算の部分の有用性を検証するための一部分の実装しか行っておらず、未実装部分の設計をさらに詳細且つ具体的に行い、提案したポリシーシステム全体を構築して、実際のネットワークに適用可能かどうかを検証していくことが必要と考える。特に、以下の点については再度深く検討する必要がある。

- ポリシー表現
- ポリシーコンフリクト検出
- メタフィルタリング記述

また、IETF 等のポリシーベースの管理システムについての動向も注目し、取り入れていく必要がある。

謝辞

本研究を進めるにあたり、指導教官である篠田陽一助教授には、様々な助言、日頃より適切な指導をたまわり深く感謝致します。また、有益な議論をして頂きました篠田研究室の皆様に感謝致します。

横河電機株式会社には、本研究を行う機会を与えて頂き、特に向井 正和様、星 哲夫様、太田 一史様、内藤 誠一様、中島 一様をはじめとする多くの方には様々な御援助を頂き深く感謝致します。

最後に、本研究を進める上であらゆる面で私を支えてくれた妻と子供たちに心からの感謝の気持を贈ります。

参考文献

[1] Filter Language Compiler. http://coombs.anu.edu.au/ avalon/flc.html.

[2] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Research in Security and Privacy, Oakland, CA, May 1997. IEEE Computer Society Press.

[3] Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and Elisa Bertino. A unified framework for enforcing multiple access control. In *Proceedings ACM SIGMOD International Conference on Management of Data*, May 1997.

[4] D. Jonscher and K. R. Dittrich. Argos – A configurable access control system for interoperable environments. In *IFIP WG 11.3 Ninth Annual Working Conference on Database Security*, Rensselaerville, NY, 1995.

[5] Hugh Mahon, Yoram Bernet, and Shai Herzog. Requirements for a Policy Management System. Internet Draft draft-ietf-policy-req-01.txt, October 1999. Work in progress...

[6] D. A. Marriott, M. S. Sloman, and N. Yialelis. Management policy service for distributed systems. Technical Report DoC 95/10, Department of Computing, Imperial College, London, October 1995.

[7] J. Bret Michael, Edgar H. Sibley, and David C. Littleman. Integration of formal and heuristic reasoning as a basis for testing and debugging computer security policy.

In *Proceedings on the 1992– 1993 ACM SIGSAC on New security paradigms workshop*, pp. 69–75, 1993.

[8] J. D. Moffett, D. Jonscher, and J. A. McDermid. The policy obstacle course. SCHEMA/York/93/1 version 1, July 1993.

[9] J. D. Moffett and M. S. Sloman. The representation of policies as system objects. In *Proceedings of the Conference on Organizational Computer Systems*, Atlanta, Georgia, November 1991. ACM SIGOIS.

[10] Jonathan D. Moffett. *Specification of Managmenet Policies and Discretionary Access Control*, chapter 17. Depart. of Computer Science, University of York, June 1994.

[11] Jonathan D. Moffett and Morris S. Sloman. The representation of policies as system objects. In *Conference on Organizational Computing Systems*, Objects, pp. 171–184, 1991.

[12] Jonathan D. Moffett and Morris S. Sloman. Policy conflict analysis in distributed system management. *Journal of Organizational Computing*, April 1993.

[13] Jonathan D. Moffett and Morris S. Sloman. Policy conflict analysis in distributed system management. *Journal of Organizational Computing*, Vol. 4, No. 1, pp. 1–22, 1994.

[14] B. Moore, E. Ellesson, and J. Strassner. Policy Framework Core Information Model – Version 1 Specification. Internet Draft draft-ietf-policy-core-info-model-03.txt, January 2000. Work in progress...

[15] Ramesh V. Peri. *Specification and Verification of Security Policies*. PhD thesis, University of Virginia, January 1996.

[16] M. S. Sloman, J. D. Moffett, and K. P. Twidle. *Domino Domains and Policies: An introduction to the Project Results*, Vol. Network and Distributed Systems Management, chapter 17. Addison-Wesley, 1994.

[17] M. Stevens, W. Weiss, H. Mahon, B. Moore, J. Strassner, G. Waters, A. Westerinen, and J. Wheeler. Policy Framework. Internet Draft draft-ietf-policy-framework-00.txt, September 1999. Work in progress...

[18] J. Strassner, E. Ellesson, B. Moore, and Ryan Moats. Policy Framework LDAP Core Schema. Internet Draft draft-ietf-policy-core-schema-06.txt, November 1999. Work in progress...

付録 A

プログラムの構成

1. 初期設定ファイルの読み込み

   -f : ネットワーク構成及び属性初期値設定

   -c : 初期設定コマンド

2. 属性評価式の設定
3. 標準入力からのコマンドの読み込みと結果の出力

コマンド 文法解説

コマンド名	解説
location	設置場所設定
genpe	物理機器生成
genpp	物理ポート設定
linkpp	物理ポート接続設定
genve	論理機器生成
linkvp	論理ポート接続設定
putattr	論理ポート管理属性/接続属性/サービス属性設定
genservice	サービス属性定義
genassets	物理ポート所属属性生成
putassets	物理ポート所属属性設定
genadmingr	物理ポート管理グループ属性生成
putadmingr	物理ポート管理グループ属性設定
attrfilter	属性フィルタリング設定
defaultfilter	デフォルト属性フィルタリング設定
addmatrix	属性評価マトリックス設定
delmatrix	属性評価マトリックス削除
delallmatrix	属性評価マトリックス全削除
dump	ネットワーク情報をダンプするコマンド
show	各種登録情報の表示コマンド
help	コマンドヘルプ
run	属性計算スタート
acchk	属性ポリシー違反検査
acgen	属性アクセスコントロール規則導出

表 A.1: コマンド解説

コマンド引数一覧

コマンド名	引数
location	名前 フラグ
genpe	名前 タイプ 場所
genpp	親 PE 名 MAC アドレス IP アドレス
linkpp	親 PE 名 index 親 PE 名 index
genve	名前 タイプ 親 PE 名
genvp	親 VE 名
linkvp	親 VE 名 index 親 VE 名 index
putattr	親 VE 名 index amdin 属性値
putattr	親 VE 名 index connection 属性値
putattr	親 VE 名 index trust 属性値
putattr	親 VE 名 index service 属性値
genservice	名前
genassets	名前
putassets	PE 名 所属名
genadmingr	名前
putadmingr	PE 名 管理グループ名

表 A.2: プロトタイプコマンド引数一覧 (ネットワーク構成記述と属性初期設定)

コマンド名	引数
attrfilter	サービス属性名 VE 名 index index (pass|block)
defaultfilter	VE 名 index index (pass|block)
addmatrix	サービス属性名 (allow|deny) サービス属性値所属属性値 管理グループ属性値 ロケーション属性値接続属性値 管理属性値 信頼度属性値
delmatrix	サービス属性名 (allow|deny) サービス属性値所属属性値 管理グループ属性値 ロケーション属性値接続属性値 管理属性値 信頼度属性値
delallmatrix	サービス属性名

表 A.3: プロトタイプコマンド引数一覧 (フィルター設定と属性評価マトリックス設定)

コマンド名	引数
run	[VE 名]
step	[VE 名]
acchk	
acgen	

表 A.4: プロトタイプコマンド引数一覧 (属性計算実行、アクセスコントロール規則生成等)

コマンド名	引数
dump	pe PE 名
dump	ve VE 名
dump	penet [PE 名]
dump	venet [VE 名]
dump	service [PE 名]
dump	trust
dump	location
show	service
show	assets
show	admingr
show	matrix [サービス属性名]
help	

表 A.5: プロトタイプコマンド引数一覧 (状態ダンプ、内部データ照会)

付録 B

プロトタイプ：ネットワーク構成記述設定ファイル例

```
location "OutSite" LF_UNSECURE LF_OUTAREA ;
location "InSite1" LF_SECURE LF_INAREA LF_PRIVATE1 ;
location "InSite2" LF_UNSECURE LF_INAREA LF_PRIVATE4 ;
location "InSite3" LF_SECURE LF_INAREA LF_PRIVATE2 ;

genservice "WWW client-" 0 ;
genservice "WWW client+" 1 ;
genservice "WWW Server" 1 ;

genpe "router1" PE_ROUTER "InSite1" ;
genpp "router1" 0:e0:2b:80:89:0 150.65.32.10 ;
genpp "router1" 8:0:20:b4:2:6 150.65.190.9 ;
genpp "router1" 8:0:20:7d:c6:7c 150.65.7.10 ;

genpe "hub1" PE_HUB "InSite1" ;
genpp "hub1" 8:0:20:b3:3e:19 150.65.190.20 ;
genpp "hub1" 8:0:20:a6:cf:27 150.65.190.21 ;
genpp "hub1" 8:0:20:7d:b3:df 150.65.190.22 ;
genpp "hub1" 8:0:20:a7:23:e4 150.65.190.23 ;

genpe "host1" PE_HOST "InSite1" ;
genpp "host1" 8:0:20:b3:3d:e9 150.65.190.1 ;

genpe "host2" PE_HOST "InSite1" ;
genpp "host2" 8:0:20:9c:24:c5 150.65.190.2 ;
```

```
genpe "host3" PE_HOST "InSite1" ;
genpp "host3" 8:0:20:b3:3c:f0 150.65.190.3 ;

genpe "host4" PE_HOST "InSite1" ;
genpp "host4" 8:0:20:b3:4d:70 150.65.7.4 ;

genpe "hostx" PE_HOST "OutSite" ;
genpp "hostx" 8:0:20:7d:ff:e8 202.249.11.99 ;

linkpp "router1" 0 "hostx" 0 ;
linkpp "router1" 1 "hub1" 0 ;
linkpp "router1" 2 "host4" 0 ;
linkpp "hub1" 1 "host1" 0 ;
linkpp "hub1" 2 "host2" 0 ;
linkpp "hub1" 3 "host3" 0 ;

genve "verouter1" VE_ROUTER "router1" ;
genvp "verouter1" ;
putattr "verouter1" 0 admin PA_ADMIN ;
putattr "verouter1" 0 connection PA_CONOUT ;
putattr "verouter1" 0 trust PA_UNTRUSTED ;
genvp "verouter1" ;
putattr "verouter1" 1 admin PA_ADMIN ;
putattr "verouter1" 1 connection PA_CONIN ;
putattr "verouter1" 1 trust PA_COMPLETE ;
genvp "verouter1" ;
putattr "verouter1" 2 admin PA_ADMIN ;
putattr "verouter1" 2 connection PA_CONIN ;
putattr "verouter1" 2 trust PA_COMPLETE ;

genve "vehub1" VE_HUB "hub1" ;
genvp "vehub1" ;
putattr "vehub1" 0 admin PA_ADMIN ;
putattr "vehub1" 0 connection PA_CONIN ;
putattr "vehub1" 0 trust PA_MARGINAL ;
genvp "vehub1" ;
```

```
putattr "vehub1" 1 admin PA_ADMIN ;
putattr "vehub1" 1 connection PA_CONIN ;
putattr "vehub1" 1 trust PA_MARGINAL ;
genvp "vehub1" ;
putattr "vehub1" 2 admin PA_ADMIN ;
putattr "vehub1" 2 connection PA_CONIN ;
putattr "vehub1" 2 trust PA_MARGINAL ;

genve "vehub2" VE_HUB "router1" ;
genvp "vehub2" ;
putattr "vehub2" 0 admin PA_ADMIN ;
putattr "vehub2" 0 connection PA_CONIN ;
putattr "vehub2" 0 trust PA_MARGINAL ;
genvp "vehub2" ;
putattr "vehub2" 1 admin PA_ADMIN ;
putattr "vehub2" 1 connection PA_CONIN ;
putattr "vehub2" 1 trust PA_MARGINAL ;

genve "vehub3" VE_HUB "router1" ;
genvp "vehub3" ;
putattr "vehub3" 0 admin PA_ADMIN ;
putattr "vehub3" 0 connection PA_CONIN ;
putattr "vehub3" 0 trust PA_MARGINAL ;
genvp "vehub3" ;
putattr "vehub3" 1 admin PA_ADMIN ;
putattr "vehub3" 1 connection PA_CONIN ;
putattr "vehub3" 1 trust PA_MARGINAL ;
genvp "vehub3" ;
putattr "vehub3" 2 admin PA_ADMIN ;
putattr "vehub3" 2 connection PA_CONIN ;
putattr "vehub3" 2 trust PA_COMPLETE ;

genve "vehub4" VE_HUB "hub1" ;
genvp "vehub4" ;
putattr "vehub4" 0 admin PA_ADMIN ;
putattr "vehub4" 0 connection PA_CONIN ;
```

```
putattr "vehub4" 0 trust PA_MARGINAL ;
genvp "vehub4" ;
putattr "vehub4" 1 admin PA_ADMIN ;
putattr "vehub4" 1 connection PA_CONIN ;
putattr "vehub4" 1 trust PA_MARGINAL ;

genve "vehost1" VE_HOST "host1" ;
genvp   "vehost1" ;
putattr "vehost1" 0 admin PA_ADMIN ;
putattr "vehost1" 0 connection PA_CONIN ;
putattr "vehost1" 0 trust PA_MARGINAL ;
putattr "vehost1" 0 service "WWW client+" ;

genve "vehost2" VE_HOST "host2" ;
genvp   "vehost2" ;
putattr "vehost2" 0 admin PA_ADMIN ;
putattr "vehost2" 0 connection PA_CONIN ;
putattr "vehost2" 0 trust PA_MARGINAL ;
putattr "vehost2" 0 service "WWW client+" ;

genve "vehost3" VE_HOST "host3" ;
genvp   "vehost3" ;
putattr "vehost3" 0 admin PA_ADMIN ;
putattr "vehost3" 0 connection PA_CONIN ;
putattr "vehost3" 0 trust PA_MARGINAL ;
putattr "vehost3" 0 service "WWW client+" ;

genve "vehost4" VE_HOST "host4" ;
genvp   "vehost4" ;
putattr "vehost4" 0 admin PA_ADMIN ;
putattr "vehost4" 0 connection PA_CONIN ;
putattr "vehost4" 0 trust PA_COMPLETE ;
putattr "vehost4" 0 service "WWW Server" ;

genve "vehostx" VE_HOST "hostx" ;
genvp   "vehostx" ;
```

```
putattr "vehostx" 0 admin PA_NOADMIN ;
putattr "vehostx" 0 connection PA_CONOUT ;
putattr "vehostx" 0 trust PA_UNTRUSTED ;
putattr "vehostx" 0 service "WWW client-" ;

linkvp "verouter1" 0 "vehostx" 0 ;
linkvp "verouter1" 1 "vehub2"  0 ;
linkvp "verouter1" 1 "vehub3"  0 ;
linkvp "vehub2"    1 "vehub1"  0 ;
linkvp "vehub3"    1 "vehub4"  0 ;
linkvp "vehub3"    2 "vehost4" 0 ;
linkvp "vehub1"    1 "vehost1" 0 ;
linkvp "vehub1"    2 "vehost2" 0 ;
linkvp "vehub4"    1 "vehost3" 0 ;
```