コンピュータセキュリティ　10－28
（2000．　7．25）

組織間情報流通へのPOLICYCOMPUTINGアーキテクチャ適用の検討

○小熊慶一郎、中村逸一、西尾秀一、土屋茂樹、坂田祐司、菅野政孝、曽根岡昭直
(株)NTTデータ
ogumak@nttdata.co.jp

あらまし

イントラネットをポリシーベースで制御するアーキテクチャである**POLICYCOMPUTING**では、ポリシーを４つの階層に分類している。**POLICYCOMPUTING**はセキュリティ、サービス品質(QoS)、システム管理の各分野を対象としているが、本論文では、特に情報セキュリティの分野についてフォーカスし、各階層のポリシーを電子的に定義する方式について、DEN(Directory Enabled Network)による方式と、XML(Extensible Markup Language)を用いた方式について検討を行った。

また、異なったポリシーを持つ組織間で情報を共有する際の課題に触れ、それを解決するサービスである「ポリシーネゴシエータ」について、その機能、及び実現までの要件を提示する。

和文キーワード　　ポリシー、ディレクトリ、XML

Applying the POLICYCOMPUTING architecture to information exchange

Keiichiro Oguma, Itsukazu Nakamura, Shuichi Nishio, Shigeki Tsuchiya,
Yuji Sakata, Masataka Sugano, Terunao Soneoka

NTT DATA Corporation
ogumak@nttdata.co.jp

Abstract

The POLICYCOMPUTING architecture which controls intranet services and environments based on its organization's policy classifies the policy into 4 layers. In this paper, we focus on the information security, and evaluated DEN (Directory Enabled Network) style policy definition method and the definition method in XML (Extensible Markup Language).

Also, we've pointed out the issues which occur during information exchange between organizations with different policies, and proposed the "Policy Negotiator" service which will solve the issues.

key words　policy, directory, XML

1. はじめに

情報システムにおけるセキュリティの重要性が高まる中、セキュリティコンポーネント自身の機能ばかりでなく、セキュリティコンポーネントをどう組み合わせ、運用するかという情報セキュリティポリシーへの関心が高まりつつある。

我々はこれまでポリシーベースシステムのアーキテクチャとして POLICYCOMPUTING™ を提案するとともに[1]、このアーキテクチャを情報セキュリティポリシーに基づくシステム制御へ適用する方式についても検討してきた[2][3]。

本論文では、単一組織の情報セキュリティポリシーを想定したポリシー定義方式に関する検討結果を報告すると共に、POLICYCOMPUTING 適用の対象を拡大し、ポリシーの異なる組織間におけるセキュアな情報流通を実現する際に必要となる機能について考察する。

2. POLICYCOMPUTING について

POLICYCOMPUTING は、情報システム及びその運用環境を、ポリシーベースで制御するアーキテクチャである。まず、このアーキテクチャについて、「ポリシーとは何か」、「ポリシーベースシステム構築の方法」、「対象領域」、「構成要素」についてその概要を述べる。

2.1. ポリシーの分類

一般にポリシーと言っても、非常に抽象的な概念から具体的な機器の設定まで、さまざまな捉え方がある。我々はこのポリシーを以下の 4 階層に分類し、上位のポリシーが下位のポリシーにブレークダウンされていくというモデルを規定した。

(1) マスターポリシー
　情報システムを持つ組織の方針・理念など抽象的な概念を表す。企業では経営方針やビジョン、法律に於いては憲法に相当するレイヤである。

(2) ガイドライン
　情報システムを持つ組織全体に普遍的な規定を行うレイヤである。企業に於いては就業規則や情報管理規則、法律に於いては民法や刑法に相当するレイヤである。

(3) ローカルポリシー
　情報システム及びその環境の中で、独立したサービスに関する規定を行うレイヤである。企業に於いては各サービスやアプリケーションの利用ルール等、法律に於いては警察法等の各法に相当するレイヤである。

(4) 個別ポリシー
　具体的に情報システムやその環境を構成するコンポーネントの設定ルールとなるレイヤである。Firewall のフィルタリングポリシーなどがこれに相当する。

2.2. ポリシーの電子化及び関連づけ

POLICYCOMPUTING では、規定されたポリシーのうち、ローカルポリシーまでを電子化し、ポリシーサーバに保存する。情報システム及びその環境を構成するコンポーネントは、ポリシーエージェントを用いてポリシーサーバから適切なポリシーを読み出し、自らのコンポーネントに適した個別ポリシーに変換したのち、そのポリシーに従ったサービスの提供を行う。

ポリシーサーバ内において、ローカルポリシーはガイドラインへ、ガイドラインはマスターポリシーへと関連づけられており、「なぜそのようなポリシーとなっているのか」を容易に把握することを可能とする。

2.3. 対象領域

POLICYCOMPUTING は大きく分けて以下の 3 つの領域を対象とするが、本論文では主として(1)のセキュリティにフォーカスしている。

(1) セキュリティ
　Firewall やアクセス制御など情報システム自体をはじめに、入退室管理など、情報システムの環境も含めて対象とする。これは、同じレベルのセキュリティを実現するにあたっても、「入退室管理を厳しく、情報システムにおける認証を簡易にする」場合と、「入退室管理を簡易にし、情報システムにおける認証を厳しくする」場合とに対応するためである。

(2) サービス品質(QoS)
　帯域制御や負荷制御などを対象とする分野であり、現在ポリシーベース制御に関する研究がもっとも進められている分野である。

(3) システム管理
　IP アドレスの利用やシステムの可用性などを対象とする分野である。

2.4. 構成要素

POLICYCOMPUTING は、以下のようなコンポーネントによって構成され、ポリシーベースのシステム構築と運用を支援するソリューションとして提供される。

(1) 雛形ポリシー
　ポリシーを作成する際のベースとするもので、目的に応じて数種類用意される。

(2) ポリシー策定ツール
　例えば、組織内情報の不正利用対策としても、抑止策(罰則の強化等)と防止策(アクセス制限等)のバランス、また、同じ防止策の中でも、入退室管理を厳しくし、情報システムへのアクセスを容易にするというケースに対して、入退室管理は簡便に、情報システムへのアクセスを厳

しくするというケースも選択し得るなど、実現のための選択肢は多岐にわたるので、それらのバランスをとるためのツール。

(3) ポリシーサーバ
電子的に定義されたポリシーを保存する装置。ポリシーサーバとしては現在 LDAP サーバが注目されているが、POLICYCOMPUTING でも LDAP サーバをポリシーサーバとして使用する。

(4) ポリシー管理ツール
ガイドラインレベルのポリシーに基づいてローカルポリシーを定めたり、ローカルポリシーと資源の関連づけを行うためのツール。

(5) ポリシーエージェント
ポリシーサーバに格納されているポリシー(ローカルポリシー)を読み出して、各機器の個別ポリシーに展開するエージェント。

(6) ポリシー監査・監視ツール
ポリシーサーバに格納されているポリシーに基づき、定期的、もしくは常時情報システム及びその環境の監査・監視を行うツール。

3. ポリシーの定義

本章では、情報セキュリティポリシーに対してこのアーキテクチャを適用し、ポリシー定義方式に関する以下の検討を行った。

3.1. ポリシーサーバ

今回ポリシーサーバとして使用する LDAP サーバは、データ構造に関して以下のような特徴を持つ。

(1) データの単位
LDAP サーバ内のデータはエントリを単位として保存され、エントリはツリー状に構成される。各エントリは一つ以上の属性を持ち、それぞれの属性は一つ以上の属性値を持つ。

(2) スキーマ
各エントリがどのような属性を持つかについてはオブジェクトクラスによって規定され、各エントリは一つ以上のオブジェクトクラスを持つことにより、どのような属性を持つかなどが決定される。
このように LDAP サーバ内のデータ規則(データ構造や属性のタイプ、属性へのアクセス等)について定めたものをスキーマと呼び、LDAP サーバのデータはこのスキーマに従って操作される。

(3) エントリ間の関連づけ
LDAP サーバにおいて特定のエントリをアドレスするためには DN(Distinguished Name)を用い、「cn=oguma,o=nttdata,c=jp」といった形で表される。

(4) 属性値の順番
エントリ内の属性は複数の属性値を持つことができるが、この順番は実装依存となっている。また、異なる属性の属性値同士を関連づけることができないので、結果的に属性値のセットを複数格納したい場合には独立したエントリとして格納するか、もしくは複数の属性値を独自の構造で単一の属性値として格納する必要がある。

3.2. DEN におけるポリシー定義

DMTF(Distributed Management Task Force)によって標準化されている DEN(Directory Enabled Network)においても、LDAP サーバをポリシーサーバとして利用している。DEN では、以下のような形でポリシーを表現している(図 1)。

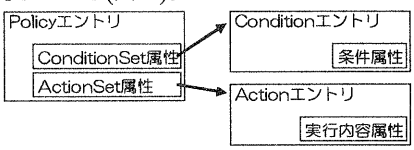

図 1 DEN におけるポリシー定義構造

DEN は現在帯域制御に関する部分を中心に標準化作業が進められているが、DEN 自体は帯域制御のみにとどまるものではなく、セキュリティの分野までも対象範囲としている。そこで、まず我々はこの DEN を標準として尊重し、適用可能な部分は適用して情報セキュリティポリシーを表現することを試みた。

3.3. マスターポリシーの定義

情報セキュリティポリシーの中でも、もっとも抽象的な部分であるマスターポリシーの定義において、我々はマスターポリシーを記述するための要件を以下のように定義した。

(1) 自然言語による記述
マスターポリシーは方針や理念を表すものであり、自然言語としての記述が可能であることが必要とされる。

(2) ガイドラインとの関連づけ
下位ポリシーであるガイドラインを規定する際に、各ガイドラインがマスターポリシー内のどのポリシーを根拠として制定されたかについて関連づけを行うことができる必要がある。

以上の要件をふまえて DEN のポリシー定義方式を検討すると、DEN の定義方式は機器に近いレベルでのポリシー定義(我々の分類によれば「ローカルポリシー」)であり、マスターポリシーの定義に用いるには適していないということがわかる。

また、上記の要件を鑑みると、XML(Extensible Markup Language)が適しているということは容易に導出することができる。今回この XML を利用して、以下のようにマスターポリシーを定義した。

```
<?xml version="1.0"?>
<!DOCTYPE accesspolicy SYSTEM "masterpol
icy1.0.dtd" >
```

```
<masterpolicy>
<date>2000 年 7 月 24 日</date>
<revision>1.0</revision>
...
<chapter num="5.3">
<title>情報管理</title>
<section>
<title>3.2 プライバシー情報の扱い</title>
<contents>
<p>  プライバシー情報を重要情報として扱う。</p>
</contents>
...
</masterpolicy>
```

3.4. ガイドラインの定義

組織内で普遍的な規定であるガイドラインについては、以下のように記述方式の検討を行った。

情報セキュリティにおいて、ガイドラインで規定する項目としては、取り扱い種別(「秘密」、「社外秘」など)と、その取り扱い方(開示範囲や暗号化等)の定義などがある。取り扱い種別それぞれに対し、情報の取得・生成、保管、流通、利用、破棄のそれぞれのフェーズを「条件」として関連づけ、さらに、それぞれの条件に対応する「動作」を関連づけることで DEN のポリシー定義方式に近い形でのポリシー定義を行うことができる。(図 2)。

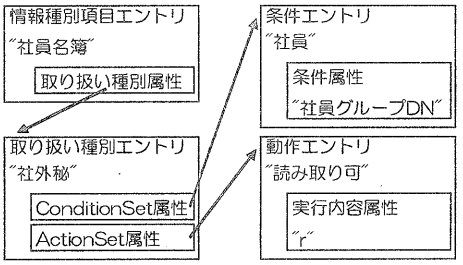

図 2 ガイドラインの定義構造

ここで新しくスキーマに追加されるデータ構造は以下のようになる。

(1) 情報種別項目エントリ

「人事情報」や「社員名簿」など情報が何であるかを示すエントリ。取り扱い種別エントリへのポインタ(DN)を持つ。

(2) 取り扱い種別エントリ

「社外秘」「秘密」など、その情報をどのように取り扱うかを示すエントリ。条件エントリと動作エントリへのポインタ(DN)を持つ。結果として、条件エントリが示す条件に該当した場合には、動作エントリが示す動作を行うというポリシーを表現する。

(3) 条件エントリ

ポリシーが適用される条件を示すエントリで、例えばアクセス制御であれば、「アクセスする人/グループ」などとなる。

(4) 動作エントリ

実行する動作を示すエントリで、例えばアクセス制御であれば「読み取り許可」などの情報が格納される。 情報セキュリティにおいて、ガイドラインで規定する項目としては、情報(「秘密」、「社外秘」など)と、その取り扱い方(開示範囲や暗号化等)などがある。

ここで、情報種別項目と取り扱い種別を別に定義したのは、一つの情報種別に対して、取り扱い種別以外のポリシーも割り当てることができるようにするためである。例えば、帯域制御ポリシーにおける帯域の優先度種別を関連づけることにより、きめ細かな回線帯域の制御も可能となる。

3.5. ローカルポリシーの定義

独立したサービスの動作を規定するローカルポリシーの規定については、以下のように検討した。

ローカルポリシーでは、実際の情報システムなどに関する具体的な規定を行うものであるが、この際、上位ポリシーであるガイドラインとの関連づけ定義を行うようにする。

例えば情報へのアクセス制御であれば、情報サービスが持つ情報がそれぞれガイドラインで規定する取り扱い種別のどれに相当するかを規定することで、結果的にある情報がどのように取り扱われるべきかを決定することができるようになる。(図 3)。

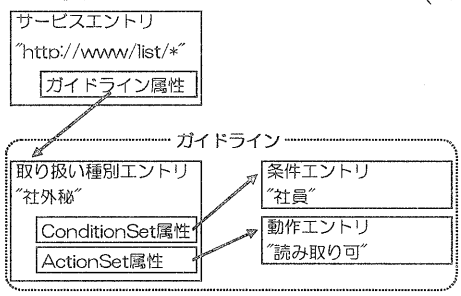

図 3 ローカルポリシーの定義構造

ローカルポリシーを、DEN 同様にエントリを単位として規定するにあたり、以下のようなデータ構造を新しくスキーマに追加した。

(1) サービスエントリ

Web サーバやグループウェアサーバなど、情報システム及びその環境に於いてサービスを提供するエンティティを表現するためのエントリであり、情報種別項目エントリ、または取り扱い種別エントリへのポインタ(DN)を持つ。

3.6. DEN 式ポリシー定義の問題点

以上のように、基本的に DEN を規範としてポリシーを規定し、実際に Web のアクセス制御へと実装を行ったところ、以下のような問題点が明らかとなった。

(1) ポリシーへのアクセスの煩雑さ

一連のポリシーが複数のエントリによって表現されているため、一つのポリシーを解釈するまでに数多くのLDAPアクセスが必要となる。

(2) 一覧性の欠如

一つの条件、及び実行内容の記述に多くのエントリを必要とするため、データの一覧性に欠ける。また、一覧性を高めるための標準的な表現形式が無い。

(3) 編集の複雑化

例えば条件エントリは、複数の取り扱い種別エントリから参照されている可能性があるため、編集する際には細心の注意を必要とする。

(4) 優先順位を規定する方式の複雑さ

複数の条件文を記述する際には、順列に関する情報を保存する必要があるが、属性が複数の属性値を取る際、LDAPはその順番を実装依存としている。

DENではPriorityという数値属性を別に定義し、優先順位を表現しているが、編集/解釈において必要以上に複雑な処理が要求される。

(5) 一貫性の欠如

LDAPではトランザクション処理的な機能は規定されていないため、複数のエントリにまたがったデータに対する一貫性の保証が行えない。

(6) スキーマ依存度が高い

ポリシー定義がスキーマに依存する割合が高いため、例えば新しい動作を定義しようとすると、スキーマを変更する必要が出てくる。スキーマはLDAPサーバ全体に影響を与えるため、ポリシーの変更がスキーマに与える影響は最小限にとどめるべきである。

実装した結果から考察するに、DENにおけるポリシー定義はあまり多くの、もしくはあまり複雑なポリシーを規定することは想定していないと考えざるを得ない。

3.7. 異なるポリシー定義方式の検討

そこで、異なるポリシー定義方式について検討を行うこととした。検討に際し、我々は以下のような要件を定義した。

(1) 標準的な記述方式の採用

可能な限り独自の記述方式を排除し、標準的な方式を採用する。

(2) 取り扱いの容易性の確保

ポリシーを解釈/表示/編集する際に、容易に取り扱えるようにする。

(3) LDAPサーバとの親和性の確保

ポリシー定義をLDAPサーバに格納するかという点については再検討の必要があるが、ポリシーを適用する対象の一部であるユーザ情報はLDAPサーバを使用することが一般化しているので、LDAPサーバとの親和性という条件は排除できない。

以上のような条件にもとづいて候補を検討したところ、Java言語とXMLの二つを考えることができた。以下、この二つについて考察を行うこととする。

3.7.1. Java言語

(1) 標準的であるか

Java言語は現在多くの局面で使用されるようになってきており、標準という意味では問題がない。

(2) 取り扱いの容易性

ポリシーの解釈に於いては、Javaクラスを読み出してJava VM(Virtual Machine)上で実行させるだけであるので非常に容易である。しかし、表示と編集においては、Java言語をそのまま使用するため、取り扱いが容易とは言えない。

(3) LDAPサーバとの親和性の確保

Java言語においてLDAPサーバ内のデータを扱う際にはSunの提供するJNDI、もしくはNetscapeの提供するLDAP JDKを使用する必要があるが、いずれのインタフェースを使用した場合も親和性が高いとは言い難い。

3.7.2. XML

(1) 標準的であるか

XMLは広く使われるようになってきており、XMLパーサも普及しつつある。しかし、ポリシーにおける条件文を定義するに適切なDTD(Document Type Definition)の標準は無く、そのDTDについては独自に規定する必要がある。ただし、汎用的なXMLパーサをそのまま利用できるという意味では標準と言うことができる。

(2) 取り扱いの容易性

XMLパーサを使用することで、容易にポリシーの解釈を行うことができる。また、表示についてもXSL(Extensible Style Language)を使用することで容易に実現できる。編集においても、XMLエディタを使用することで、DTDに基づいてポリシーを定義していくことが可能なので、容易に取り扱うことができると言える。

(3) LDAPサーバとの親和性の確保

XMLではURLベースで情報と関連づける仕組みがあり、また、LDAPサーバ内の情報はURLにより表現可能であるので、LDAPサーバとの親和性についても問題は無い。

3.7.3. 評価

以上のようにJava言語とXMLを比較すると、XMLがポリシー定義には優れていると言うことができる。

次に、このXMLを用いたポリシーの記述が、エントリを用いてポリシーを記述した際に明らかとなった問題を解決することができているかについ

て検証を行うと、それぞれ以下のようになる。

(1) ポリシーへのアクセスの煩雑さ
　一つのポリシー定義を一つのドキュメントとして定義でき、また、XMLパーサを用いることで解釈も比較的容易に行うことができる。

(2) 一覧性の欠如
　XSL を使用することで容易に表示することができる(前述)。

(3) 編集の複雑化
　XMLエディタを使用することで容易に編集することができる(前述)。

(4) 優先順位を規定する方式の複雑さ
　記述の順番で優先順位を表現できる。

(5) 一貫性の欠如
　一つのポリシー定義を一つのデータとして格納することにより、ポリシー内での一貫性が保たれる。

(6) スキーマ依存度が高い
　スキーマと独立したDTDによってポリシーの構造が規定されるため、新たなポリシーを利用する場合でもスキーマに影響を与えない。

以上のように、XMLはポリシーを記述する手段として優れていると考えることができる。

4. ポリシーの編集

引き続き、XMLで記述されたポリシーを編集するためのツールにつき、以下のように検討を行った。

4.1. ポリシー編集ツールのベース

XML で記述されたポリシーを編集するツールのベースとして、一般の XML エディタを利用することができる。

現在の XML ではタグに挟まれた領域にどのような値(例えば URL)を設定可能かの定義ができないため、利便性に難があるが、現在標準化作業中である XML Schema を適用することで、このような問題も解決すると考えられる。

4.2. ポリシー関連づけチェック機能

ポリシー編集時に、そのポリシーの根拠となる上位レイヤのポリシーを容易に参照できる必要がある。根拠となるポリシーを参照しつつポリシーを定義することで、上位ポリシーからの逸脱を防止することが可能となる。

4.3. 変更したポリシーの通知

LDAP サーバのポリシーが変更されたとき、LDAP プロトコルではその変更をクライアントに通知する手段がない。変更されたポリシーをポリシーエージェントに伝えるにあたっては、以下のような手段を考えることができる。

(1) ポーリング方式
　ポリシーエージェント、もしくはその代理となるエンティティが定期的に LDAP サーバの変更されたエントリを検索し、変更があった部分について個別ポリシーの設定を変更する方式。
　標準的な LDAP プロトコルのみで構築できるため、ポリシーの変更が頻繁には行われず、ポリシーが変更されてから設定が変更されるまでの時間的要求条件が緩い場合に適している。ポリシーエージェントの数が極端に多い場合や、ポリシーの変更をチェックする間隔が短いような場合にはネットワークへの負荷が大きくなる。

(2) LDUP 方式
　ポリシーエージェントをレプリケーション対象の LDAP サーバと見なし、ポリシーを格納している LDAP サーバのエントリが変更された際には、LDAP サーバからポリシーエージェントに対して LDUP(LDAP Duplication/ Replication/ Update Protocols)を用いてポリシーの変更を行う方式。
　効率的にポリシーの変更を通知することが可能であるが、LDUP の標準化作業が終わっていないことや、実装も進んでいない点から現時点では利用は難しい。

(3) ポリシー編集ツール通知方式
　ポリシーの編集はポリシー編集ツールを使用して行うこととなっているので、このポリシー編集ツールがポリシーの変更を通知する方式。通知に利用するプロトコルとしては、SNMP(Simple Network Management Protocol)などを利用する。
　ポリシー編集ツールは常時動作しているものではないので、ポリシー編集時点でポリシーエージェントが動作していない場合には、ポリシーの変更を通知できないという問題がある。

今回は実装の容易さからポーリング方式とポリシー編集ツール通知方式の併用という形で実装を行ったが、LDAP サーバ製品の対応を待って LDUP 方式に移行していきたい。

4.4. ポリシーへの署名

ポリシーが正しいことを検証する仕組みとして、LDAP サーバのセキュリティ(アクセス制御)に依存せず、XML電子署名を利用するということも可能である。この場合、ポリシー編集ツールが署名を行う必要がある。

5. Web サーバのアクセス制御への実装例

ここまでのように検討してきた方式を、実際に Web サーバ上の情報へのアクセス制御に適用した例について以下に示す。

今回は、Web サーバとして LDAP ベースのユーザ

認証を行うことができる Netscape Enterprise Server(以後 NES)を利用した。ポリシーエージェントは NES と同一のマシン上で動作し、LDAP サーバから取り込んだポリシーを NES の ACL(Access Control List)に変換するという仕組みを採用した。

5.1. ポリシーの定義

XML を用いて、従来「取り扱い種別エントリ」、「条件エントリ」、「動作エントリ」として記述していた内容の規定を行うと以下のようになる。

```
<?xml version="1.0"?>
<!DOCTYPE accesspolicy SYSTEM "accesspol
icy1.0.dtd" >
<accesspolicy name="秘密 (研究所管理職限り)
">
<basepolicy>ldap://ldap/cn=policy001,ou=
guidelines,ou=policies,o=nttdata,c=jp/</
basepolicy>
<rule>
<condition>
<and>
<ldapurl>ldap://ldap/cn=rd,ou=groups,o=n
ttdata,c=jp/</ldapurl>
<ldapattr><attr>title</attr><op>=</op><v
al>管理職</val></ldapattr>
</and>
</condition>
<action>
<allow_read/>
</action>
</rule>
</accesspolicy>
```

また、ポリシー全体の関連は以下の図のようになる。(図 4)。

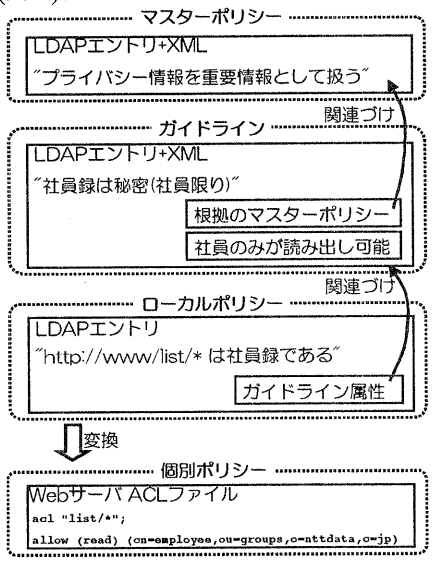

図 4 ポリシー全体の関連図

5.2. 文書種別の定義方式

NES の ACL では、ディレクトリまたはファイル名を記述するようになっており、ファイルの内容をもとにアクセスを制御することはできない。

現在、Web サーバ用ポリシーエージェントとして、XML ベースで記述された情報(ドキュメント)の内容を読み、その中に XML タグとして定義されている文書種別(「人事情報」や「社員名簿」等)に従ってアクセス制御を行うように検討している。XML ドキュメント中のタグをベースにすることで、このドキュメントが他のサービス(例えば Mail サーバ)を経由するような場合でもシームレスにポリシーベース制御を行うことができるようになる。

5.3. ポリシーエージェントの動作

ポリシーエージェントは Web サーバと同じ OS 上に存在し、定期的に LDAP サーバのポリシーが変更されていないかをチェックする。

ポリシーが変更されていた場合、または、SNMP によりポリシーの変更を通知された場合は、ポリシーエージェントは変更されたポリシーを LDAP サーバより読み込み、Web サーバの ACL(Access Control List)に変換する。

現在 Web サーバのプラグインという形でアクセス制御を実現するモジュールの検討を進めており、このモジュールを利用することで、Web サーバの ACL で表現できない、よりきめ細かな(例えば時間帯による制御)を行うことが可能となる。

6. ポリシーネゴシエータの提案

POLICYCOMPUTING が最初に目指すものは、単一のポリシーを持つ組織内において、そのポリシーを情報システム及びその環境に適切に展開していくことであるが、その次のステップとして、ポリシーの異なる組織間で、情報セキュリティを実現することを目標としている。ここでは、そこで発生する課題と、それを解決するための機能である「ポリシーネゴシエータ」について提案する。

6.1. 組織間情報流通における課題

現在、組織間で NDA(Non Disclosure Agreement)に基づいて情報流通を行うのが一般的であるが、このような状況には以下のような問題点がある。

(1) 画一的な情報管理に陥りがち

情報の種別それぞれについて組織間で共通のセキュリティレベルの概念を形成するのは困難であり、結果的に共有する情報全体に一律高いセキュリティレベルを与えてしまう。結果として、NDA が守られない、または正しい情報管理が行われないといった事態を招く。

(2) 情報管理の実態がわからない

相手組織でどのように情報管理が行われているかの実態を知ることができないため、渡した

情報が正しく管理される保証が無い。

(3) NDA 締結は時間を要する

NDA は個別に締結するため、情報を共有する相手が多くなると NDA 締結が大きな負担となる。

6.2. ポリシーネゴシエータの機能

ポリシーネゴシエータは、以上のような課題を解決する機能を持ち、ある組織から別の組織(例えば A 組織から B 組織)に情報を開示するにあたり、以下のようなステップでポリシーのネゴシエーションを行う。なお、ポリシーのネゴシエーションを行う前段階として、基本的な情報開示契約は締結されているものとする。

(1) 情報種別に基づき、A 組織の開示ポリシーと照らし合わせる。開示ポリシーは、開示できる相手、開示する際のセキュリティレベルについて記述されている。

(2) 開示ポリシーに従い、開示可能な情報につき、セキュリティレベルの変更を行う。

(3) 信頼できるセキュリティ監査機関による A 組織の監査結果を照らし合わせ、実際に A 組織がそのセキュリティレベルの情報に対して行っている情報管理を割り出し、要求レベルとする。

(4) B 組織の情報セキュリティポリシーと信頼できるセキュリティ監査機関の監査結果から、B 組織における、各セキュリティレベルで実際にどのような情報管理が行われているかを割り出す。

(5) A 組織の要求レベルと B 組織の情報管理の実態を照らし合わせ、B 組織のどのセキュリティレベルならば A 組織の要求レベルをクリアできるかを判断する。

(6) 最終的に、B 組織の情報セキュリティポリシーの枠組みに割り当てられる形で情報が B 組織に渡される。

B 組織の情報管理の実態によっては、A 組織の要求するレベルをクリアできず、情報開示ができないというケースもあり得る。

以上のように、信頼できる第三者としてのポリシーネゴシエータが情報の開示を仲介することにより、以下のような利点が生まれる。

(1) 基本的な情報開示契約を結ぶのみで、迅速に情報の共有を開始できる。

(2) 自組織の情報セキュリティポリシーと別個の情報セキュリティポリシーを作成もしくは利用する必要がない。

(3) 相手組織に自組織のセキュリティポリシーを開示する必要がない。

(4) 情報の種別に応じた適切な情報管理が適用される。

(5) 信頼のおけるセキュリティ監査機関による監査結果に基づいて判断するため、相手組織内での適切な情報管理が期待できる。

6.3. ポリシーネゴシエータ実現までの課題

ポリシーネゴシエータを実現するためには、情報を流通する両組織で以下のような前提条件が必要となる。

(1) ポリシーが規定されていること

(2) ポリシーベースで情報システムが動作していること

(3) ポリシー定義をポリシーネゴシエータが解釈可能であること

(4) セキュリティ監査が実施されていること

(5) セキュリティ監査結果をポリシーネゴシエータが解釈可能であること

このうち、ポリシー定義及びセキュリティ監査結果をポリシーネゴシエータが解釈し、他の組織と比較できるようにするためには、ポリシー定義及びセキュリティ監査結果について、それぞれの持つ意味を厳密に規定することが必要とされる。

なかでも、ポリシーの定義を変更することは情報システム及びその環境に多大な影響を与えかねないので、当初のポリシー策定時から慎重に検討することが必要である。

7. おわりに

「セキュリティシステムを適切に利用する」という言葉の意味は、すなわち組織のポリシーに沿った形で利用するということに他ならない。本報告では、組織のポリシーに従った形でセキュリティシステムを制御するための基本概念、及びその実現手段である「POLICYCOMPUTING」により、その解を提示した。

更に、ポリシーの異なる組織間で情報を流通する際に発生する問題点に触れ、その解決策として「ポリシーネゴシエータ」を提案し、実現までの課題を提示することで、今後ポリシーベースシステム導入を進める際の注意点に言及した。

今後は、POLICYCOMPUTING の適用範囲の拡大、及びポリシーネゴシエータの詳細化について検討を進めていく。

[参考文献]

[1] 田中、菅野、他:情報ネットワークシステムのポリシー制御 POLICYCOMPUTING™ に関する一検討, 情処研報 99-DPS-18

[2] 菅野、坂田、他: POLICYCOMPUTING™ のセキュリティポリシーへの適用, 情処研報 99-DPS-56

[3] 坂田、小熊、他,POLICYCOMPUTING™ アーキテクチャによる情キュリティポリシーの実装と導入, 情処全大 60-5Q8