2004－CSEC－27　(17)
2004／12／20

機能マッピングによるセキュリティ対策状況の把握

松田 勝志, 岡城 純孝, 小川 隆一

NEC インターネットシステム研究所

概要

セキュリティポリシーと実際のセキュリティ設定には表現上の大きなギャップがあるため, ポリシーに基づくセキュリティマネジメントは上手く行かないことが多い. 本稿では, ネットワークシステムのセキュリティ設定に関するセキュリティポリシーを不具合なく策定して表現するためのセキュリティポリシー記述の体系と, この体系によって記述されるセキュリティポリシーとそのセキュリティポリシーをエンフォースしたい実際のネットワークのトポロジーを対応付ける機能マッピング方式について述べる. 記述体系によって, 漏れのないセキュリティポリシーを簡単に策定でき, 機能マッピングによって, セキュリティ対策の状況把握が可能になる.

Grasping the Security Management Situation by Functional Mapping

Katsushi MATSUDA, Sumitaka OKAJO and Ryuichi OGAWA

Internet System Research Laboratories, NEC

Abstract

There is a great expressive difference between a security policy and actual security parameters in security management, and consequently, policy-based security management might be difficult to accomplish. In this paper, we propose a new security policy description method that is easy and consistent, and also describe a functional mapping technique that matches a fragment of security policy with a software or hardware node in a network system. The description helps security administrators to draw up security policies without omissions. Also functional mapping helps them to grasp their security management situation and to discover some inadequate pieces of policy or nodes in the network system from the mapped situation.

1 はじめに

セキュリティポリシーを策定し, ファイアウォールやアクセス制御によるネットワークシステムのセキュリティ対策が行われているにも関わらず, 不正アクセスやワーム感染や情報漏えい等の脅威は一向に低下しない. この原因の一つにセキュリティポリシーの表現と実際のセキュリティ設定とのギャップによるポリシーエンフォースの不徹底が考えられる. 従来, このギャップはコンサルタントやセキュリティ専門家による人手に頼っており, 高コストであった. そのため, 自動化によるコスト低減と品質の均一化が望まれている.

そこで本稿では, 表現のギャップに起因するポリシーエンフォースの不徹底は, ポリシー記述体系の未整備と実際のネットワークシステムのトポロジーとの対応付けができていないことが原因であると考え, ネットワーク機器とサーバのセキュリティ設定に関するセキュリティポリシー記述体系と, この記述体系を用いたセキュリティポリシーとそのポリシーを

エンフォースしたい実際のネットワークシステムのトポロジーを対応付ける機能マッピング方式について報告する. 機能マッピングによって, セキュリティポリシーとトポロジーを対応付けることができるため, どの機器にどのようなセキュリティ設定がなされるのかというセキュリティ対策状況の把握が容易になる.

2 企業のセキュリティ対策

企業がセキュリティに取り組む場合, セキュリティポリシーを策定し, それに基づいたセキュリティ対策を実施することが多い. 更により広く BS7799[1]や ISMS[2]のようなセキュリティマネジメントの指針に従うこともある. 図１にセキュリティポリシーの階層モデルを示す.

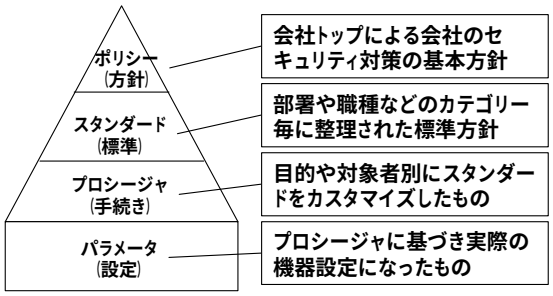

図１．ポリシー階層モデル

セキュリティポリシーの策定は, 最も抽象的な「方針」から作成し, 部門や目的に応じて順に具体化することで「標準」「手続き」を作成する. そして最後に現場のセキュリティ管理者やシステム管理者が「手続き」に基づいて機器の「設定」を作成・適用する.

2.1 セキュリティポリシー策定の問題

しかしながら, セキュリティポリシー自身がその企業の機密情報であり公開されないため, セキュリティポリシー全般に関する確立した体系が存在しない.

唯一公開されているのが, 日本ネットワークセキュリティ協会(JNSA)のセキュリティポリシーWG が発行した情報セキュリティポリシーサンプル[3]である. このサンプルは, 2 つの「方針」と 29 の「標準」から構成されており, 漏れのない完全なセキュリティポリシーを目指している. しかし, その構成や内容は体系化されていないため, 参照や利用に際しては非常に不便である. 例えば, あるネットワークに新規にサーバを設置する場合, 人的な手続きを除いても, 13 の「標準」を参照する必要がある. しかも, これらの「標準」では重複して記述されたものや微妙な言い回しの違い等が存在する.

すなわち, セキュリティポリシーを策定すること自身が困難であり, 策定したとしても簡潔で漏れや矛盾のないポリシーになっているとは限らない, という問題が存在する.

2.2 セキュリティ対策の状況把握の問題

セキュリティポリシーが策定できた場合にも問題がある. 一般に, セキュリティ対策[1]をゼロから開始することはなく, 既に何らかのセキュリティ対策が施されている場合がほとんど(アクセス制御など)である. このような場合, まずは現状のセキュリティ対策の状況把握を行った上で, 策定したセキュリティ対策との差異を明確にする必要がある. しかし, 現状のセキュリティ対策はどのようなポリシーの元で行われているのか, 実際にどのような対策がなされているのかは, セキュリティ管理者であっても曖昧な場合が多く, 把握することは困難である.

このようにセキュリティポリシーに基づくセキュリティ対策には大きな問題がある. 本稿では, これらの問題を解決するセキュリティポリシーの記述体系とセキュリティ対策状況把握の技術について述べる.

3. セキュリティポリシー記述体系

3.1. 記述体系の設計コンセプト

本稿で提案するセキュリティポリシー記述体系の設計コンセプトは, 以下の 4 点である.

(1) 標準および手続きを対象とする
(2) ネットワークシステムを対象とする
(3) 重複や矛盾などの不具合が入りにくい
(4) 機械可読である

ポリシーの表現レベルは「標準」および「手続き」とした.「方針」は非常に抽象度が高く, 記述体系に組み込む積極的な理由がないため, 対象外とした.

対象とするセキュリティの種類は, ネットワ

[1] セキュリティ機器に対して「設定」を適用(エンフォース)すること.

ークシステムにおけるネットワーク機器やサーバのセキュリティ設定に関するものに限っている．すなわち，媒体の取り扱いやセキュリティ教育や罰則等の人的セキュリティ要件やクライアント PC 等のセキュリティ設定は対象外とした．

前述の JNSA のポリシーサンプルでは，そのドキュメントを読む人の目的や状況に応じてドキュメントを分けていたため，同じ記述や似た記述が複数のドキュメントに渡り記述されている．そこで本体系では，ポリシーをセグメント単位で分類した．セグメントとは，ある目的のために明確に分離されたネットワークのゾーン，およびそれらゾーン間の境界部分である．セグメントの例としては，DMZ セグメントや LAN セグメント，そして DMZ-INT セグメント(DMZ とインターネットの境界セグメント)等がある．これらのセグメント毎にポリシーを用意しておくことによって，あるセグメントのあるハードウェアの設定やソフトウェアの設定は唯一のものとなるため，重複や矛盾の可能性を削減できる．

自然言語によるポリシーは，ポータビリティや検証に向かないため，XML を用いたセキュリティポリシー記述体系とした．これによって 4 節で述べるセキュリティ対策状況の把握が可能となる．

3.2. 機能クラス

あるポリシーにおいて記述された機能と別のポリシーで記述された機能が，同一であるか，全く別のものか，関連するものか，を識別できなければならない．本セキュリティポリシー記述体系では，これらを識別するために，機能クラスというセキュリティ機能を表現する共通語彙体系を導入する．機能クラスは，セキュリティ機能に特化した抽象クラスであり，ツリー構造で階層的に分類したものである．また，機能クラスには，セグメントの種類も含んでいる．図 2 に機能クラスの例を示す．

機能クラスは，ドット(.)がツリー構造のアークであり，ノードを重ねることにより詳細なセキュリティ機能を表現するように構成されている．例えば，function.filtering.packet という機能クラスは，パケットフィルタリング機能のことであり，function.filtering.contents という機能クラスはコンテンツフィルタリング機能のことである．更に，これらの機能クラスは function.filtering というフィルタリング機能に属するため，セキュリティ的には近い機能であることが分かる．

```
network.segment.dmz                 DMZセグメント
network.segment.internet            インターネットセグメント＝インターネット
network.segment-boundary.dmz-int    DMZとインターネットとのセグメント境界
network.segment-boundary.intra-wan  イントラネットとWANとのセグメント境界
function.router                     ルーティング機能
function.filtering.packet           パケットフィルタリング機能
function.filtering.contents         コンテンツフィルタリング機能
function.address.ipmasqarade       IPマスカレード機能
function.service.integrity          ファイル改竄防止サービス機能
function.service.job                バックグランドジョブサービス機能
function.encode.des.54              54ビットDES暗号方法機能
account.generic.root                rootアカウント
account.generic.user                一般ユーザアカウント
file.acl.readonly                   読み込み可能のファイルアクセス権
file.acl.readwrite                  読み書き可能のファイルアクセス権
file.password                       パスワードファイル
packet.forward                      パケット転送
packet.ip.tcp.httpd                 80/TCPパケット
packet.ip.udp.1434                  1434/UDPパケット
information.fingerprint             フィンガープリント情報
....
```

図 2．機能クラスの例

3.3. セキュリティポリシー記述言語

セキュリティポリシーを機能クラスを用い，セグメント別に記述できるようにしたものがセキュリティポリシー記述言語である．図 3 にセキュリティポリシー記述言語で記述したポリシーの例を示す．

```
<standard name="DMZ-001-1">
        <comment>ログを取る</comment>
        <subject>function.logging</subject>
        <enforce>add</enforce>
        <segment>network.segment.dmz</segment>
</standard>

<standard name="DMZ-002-1">
        <comment>ネットワークサービスの管理をする</comment>
        <subject>function.authorize.service</subject>
        <enforce>add</enforce>
        <segment>network.segment.dmz</segment>
</standard>
<procedure name="DMZ-002-1-1">
        <comment>必要のないネットワークサービスを起動しない</comment>
        <parent>DMZ-002-1</parent>
        <subject>service.*</subject>
        <set>disauthorize</set>
</procedure>
```

図 3．ポリシーの例

例えば，最初のポリシー(DMZ-001-1)は，DMZ セグメント(network.segment.dmz)において，ログ(function.logging)を取る(add)，という「標準」である．また，図 3 にはないが，あるポリシーと別のポリシーが同時に実現することができないことを示すタグや，同じハードウェア上の複数のソフトウェアには同時に用いられないポリシーであることを示すタグ等も用意している．

特筆すべき特徴は，あるポリシーがどのセキュリティ機器で実施すべきか書かれないこと

である．すなわち，あるセグメントにおいて実現すべきセキュリティ機能は記述するが，どのようなネットワーク機器やソフトウェアでそのセキュリティ機能を実現するかは記述しない．これによって，例えば，企業の持っている全ての DMZ に共通のセキュリティポリシーの「標準」を策定することができる．

なお，本記述言語は，セキュリティポリシー言語 SCCML[4]の上位言語にあたり，セキュリティ機器のより具体的な設定の記述は SCCML を用いる．

3.4. 自然言語による記述との比較

従来の自然言語によるポリシーと本稿で提案するポリシー記述言語を様々な観点から比較したものが表 1 である．ここで，「参照容易性」とは，利用者が目的のポリシーを参照する際の容易さである．「一貫性」とは，あるポリシーがどこに記述されていても同じ内容である必要がある場合の内容の同一性の保証である．「ポリシー自身の不整合チェック」とは，ポリシーの不備や矛盾や対象ネットワークとの乖離をチェックすることの容易さである．「維持容易性」とは，時間とともに変化する対象ネットワークの構成や利用目的への追従に伴うポリシーの修正しやすさである．「現状の対策との比較」とは，現在稼動中の対象ネットワークに施されたポリシーとの比較しやすさである．

	自然言語によるポリシー	本稿ポリシー記述
分類方法	利用者の目的や立場	セグメント単位
参照容易性	容易	比較的容易
記述方法	自然言語	XML
記述語彙	人手によって語彙の統一	機能クラスを利用
一貫性	持たせるのは困難	持たせるのは容易
記述の柔軟性	高い	低い
ポリシー自身の不整合チェック	困難	容易
維持容易性	困難	容易
パラメータへの変換	人手で変換	自動で変換可能
現状の対策との比較	人手で逐次比較	自動で比較可能

表 1. ポリシー記述方式の比較

表からも分かるように，参照容易性や記述の柔軟性など一部の観点からは従来の自然言語によるポリシー記述の方が優れている．しかし，例えば，「記述の柔軟性」を重視すると，セキュリティ対策の不均一が発生したり，意図がセキュリティ管理者に伝わらなかったり，という問題が生じることがある．すなわち，「記述語彙」や「一貫性」に優れた本記述言語の有用性は高く，2.1 節のセキュリティポリシー策定の問題を解決するものと考える．

4 機能マッピング

4.1. セキュリティ対策状況の把握

表 1 の「現状の対策との比較」は、2.2 節で示したセキュリティ対策の状況把握の問題そのものである．

一方，前節で提案したポリシー記述言語は，用いる語彙を機能クラスという抽象化されたセキュリティ機能で統一している．同様の方針で，セキュリティ機器(ソフトウェアも含む)の持つセキュリティ機能を機能クラスで表現できれば，機能クラスを用いてポリシー記述と対象ネットワークシステムのセキュリティ機器を対応付けることが可能である．更に，この対応付けの際に，ポリシー記述で規定された通りに対象ネットワークシステムのセキュリティ機器が構成されているか否かを検証することができる．すなわち，セキュリティ対策の状況が明確になる．

機能クラスを用いてポリシー記述と対象ネットワークシステムを対応付ける方式を機能マッピングと呼ぶ．機能マッピングは，セキュリティ対策状況の把握，より具体的には，ポリシーと機器の対応関係と対応付けられなかったポリシーや機器の状況を明確にする．

4.2. 利用する知識

セキュリティ機器の持つセキュリティ機能を機能クラスで表現したものがノード知識である．ノード知識は，ネットワークシステムのノード，すなわち個々のハードウェアやソフトウェアが，セキュリティ上のどのような機能を有しているのかを前述の機能クラスを用いて列挙記述した知識ベースである．図 4 にノード知識の例を示す．

```
<software canonical="ipchains" version="1.3.10">
        <function>function.router</function>
        <function>function.filtering.packet</function>
        <function>function.address.ipmasquerade</function>
</software>
<software canonical="iptables" version="1.2.7">
        <inherit canonical="ipchains" version="1.3.10"/>
        <function>function.address.snat</function>
        <function>function.address.dnat</function>
</software>
```

図 4. ノード知識の例

一方, 対象ネットワークシステムの構成を記述するのが, トポロジー記述である. トポロジー記述は, 対象ネットワークシステムのセグメント構成, ハードウェア構成, そしてハードウェア上のソフトウェア構成をそれぞれ関係付けて記述することができる. 図 5 にトポロジー記述を用いて記述したトポロジーの例を示す.

```
<topology>
    <segment name="int" type="network.segment.internet">
        <address>0/0</address>
    </segment>
    <segment name="int-dmz" type="network.segment-boundary.int-dmz">
        <address>207.100.5.233</address>
        <address>207.190.1.1</address>
        <hardware name="fw0">
            <canonical>NEC Express5800</canonical>
            <model>54wd</model>
            <nic name="eth0">207.100.5.233</nic>
            <nic name="eth1">207.190.1.1</nic>
            <software name="fw0-os">….</software>
            <software name="pf1s">
                <canonical>iptables</canonical>
                <version>1.4.3</version>
            </software>
            ・・・
    <segment name="dmz" type="network.segment.dmz">
        <address>207.190.1.1/24</address>
        <hardware name="www">
            ・・・
</topology>
```

図 5. トポロジーの例

トポロジー記述は, ポリシー記述とは反対に, あるノードがどのポリシーを実現できるか書かれていない. すなわち, ポリシーからは独立した記述となっている.

4.4. 機能マッピングと矛盾判定

機能マッピングは, 機能クラスが一致するポリシーとノードを対応付ける処理を行う. このポリシーとノードの組み合わせをタプルと呼ぶ. 機能マッピングの結果を用いて, 入力されたポリシーとトポロジーに不備や矛盾がないかを判定できる. 具体的には, 機能マッピングの結果のタプルの種類や 2 つ以上のタプルの関係によって矛盾かどうかを判定する. このタプルの種類や組み合わせによって矛盾かどうかを判定する矛盾判定式を表 2 に示す.

f はポリシーで定義されたセキュリティ機能を実現する, ¬f はそのセキュリティ機能を実現しない, をそれぞれ表す. n はノード, h はあるハードウェアを表している. また, 記号$は機能マッピングによるタプルを, 記号⊥はポリシーに指定されている互いに独立なセキュリティ機能をそれぞれ示している. 例えば, 表 2 の(3)は, あるノードがいずれのポリシーともタプルとなっていないことを表している.

番号	機能とノードの関係	矛盾
(1)	∃n(fa $ n ∧ ¬(fb $ n ∧ fa≠fb))	-
(2)	∃n(¬fa $ n ∧ ¬(¬fb $ n ∧ fa≠fb))	-
(3)	∀n(¬ ∃f(f $ n))	overpolicy
(4)	∀n(¬ ∃f(¬f $ n))	-
(5)	∀f(¬ ∃n(f $ n))	underpolicy
(6)	∃n(fa $ n ∧ fb $ n ∧ …)	-
(7)	∃n(¬fa $ n ∧ ¬fb $ n ∧ …)	-
(8)	∃n(fa $ n ∧ ¬fb $ n ∧ … ∧ fa≠fb ∧ …)	-
(9)	∃n(fa $ n ∧ ¬fa $ n)	collision
(10)	∃f(f $ n1 ∧ f $ n2 ∧ …)	-
(11)	∃f(¬f $ n1 ∧ ¬f $ n2 ∧ …)	-
(12)	∃fa∃fb(fa $ n1 ∧ fb $ n2 ∧fa⊥fb)	collision
(13)	∃fa∃fb(¬fa $ n1 ∧ ¬fb $ n2 ∧ ¬fa⊥¬fb))	collision
(14)	∃f∃h(f $ n1 ∧ f $ n2 ∧ … ∧ n1,n2,..∈h)	collision

表 2. 矛盾判定式

これらの判定式によって検出する矛盾には, Collision(ポリシー衝突)と Overpolicy(ポリシー過剰)と Underpolicy(ポリシー過少)がある(表 2 の第 3 列,「-」は矛盾なしを表す).

Collision とは, お互いに相反する一つ以上のポリシーが, ある一つのノードに対応付けられた場合を言う. 典型的な例は, 表 2 の(9)であり, 相反する機能が同じノードに対応付けられている場合である.

Overpolicy とは, ポリシーによって実現を規定しているセキュリティ機能があるにも関わらず, それを実現するノードが存在しない場合である(表 2 の(3)).

Underpolicy とは, どのポリシーとも対応付けられなかったノードが存在する場合である. すなわち, セキュリティ機能を持ったノードがインストールされているにも関わらず, そのノードの機能をどうするか(実施するかしないか)がポリシーとして定義されていない場合である(表 2 の(5)).

4.5 矛盾解消

前節の矛盾判定によって, 様々な矛盾の検出ができる. 以下, それらの矛盾を解消する方法について示す.

Collision には, 大きく分けて 3 種類の競合状態がある.

(A) 正反対のポリシー
(B) 同時に実現不可と規定されたポリシー
(C) 同じハードウェア上では実現不可と規定されたポリシー

このうち，(A)と(B)の Collision は，1 つのノードに対応するポリシーが複数という形であるため, いずれか一方のポリシーを削除することで矛盾が解消できる．(C)の Collision は，同一ハードウェア上で複数のノードに対応付けられないように, ノードを削除することで矛盾が解消できる．

Overpolicy は，ポリシーはあるが，それを実現するノードが存在しない状態であるため，この矛盾を解消するには, ポリシーを削除するか，ポリシーを実現するノードを追加する．

Underpolicy は，ノードはあるが，そのノードが持つセキュリティ機能を用いるポリシーが存在しない状態であるため, この矛盾を解消するには，ノードを削除するか，そのノードの機能を用いるポリシーを追加する．

5 機能マッピングシステム

機能マッピングによってセキュリティ対策状況の把握を行うシステムの実装を行った. 図 5 にシステムの動作画面を示す．

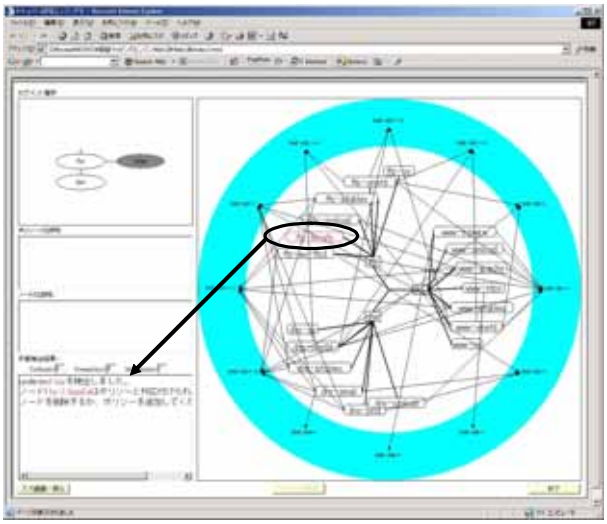

図 5. 機能マッピングシステム

システムはセグメント毎にポリシーとノードの対応関係をグラフで表現する(外周部にポリシー，内部にノードが配置されており，対応は線分で表されている)．図 5 では，システムは，機能マッピングの結果，Underpolicy であることを示している．

このように機能マッピングシステムは, ポリシーとトポロジーを入力とすると, ポリシーとノードの対応関係やポリシーの矛盾や不備, ノードの不備等のセキュリティ対策の状況を表示する．これによって，苦労して策定したセキュリティポリシーがあるにも関わらず, そのポリシーに従っていないネットワークシステムの構成になっていたり, 高価なセキュリティ対策機器を購入して設置したにも関わらず, その機器を有効活用しないポリシーになっていたり，等のセキュリティ対策の不具合を発見，解消することができる．

7 おわりに

ネットワークシステムにおける簡潔で漏れや矛盾のないセキュリティポリシーを「標準」「手続き」のレベルで策定するためのセキュリティポリシー記述体系と, 対象ネットワークシステムの既存のセキュリティ対策の状況把握を容易にし, セキュリティ管理者のセキュリティマネジメントを支援する機能マッピング方式ついて述べた. セキュリティポリシー記述体系では，セグメント単位での分類，機能クラスを用いた語彙の統一を基本とし, 機械可読性の高いポリシー記述言語を設計した. セキュリティ対策状況の把握では, 実施すべきセキュリティポリシーと現状のネットワークトポロジーを入力として, それらの対応関係を明確にすることでポリシーやトポロジーの矛盾を検出し，現状のセキュリティ対策の不具合を指摘する機能マッピングについて述べた．

本稿では，3 種類の矛盾を判定したが，実際にはセキュリティ機能間の意味的な関係やセキュリティ上のノウハウ等を矛盾判定に用いる必要がある. 今後はこれらについて検討する予定である．

参考文献

[1] BS7799: Code of practice for information security management, 1999.
[2] 財団法人日本情報処理開発協会,「情報セキュリティマネジメントシステム(ISMS)適合性評価制度」, http://www.isms.jipdec.jp/,.
[3] 日本ネットワークセキュリティ協会,「情報セキュリティポリシー・サンプル 0.92a 版」, http://www.jnsa.org/policy/guidance/, 2003.
[4] 岡城他,「セキュリティ運用管理のためのセキュリティポリシー言語 SCCML」, 情報処理学会研究報告 2004-CSEC-27(17), 2004.