2005－CSEC－31（1）
2005／12／9

マトリックス分解によるパケットフィルタリングルールの分析
－不要ルールと冗長条件ルールの検出－

松田 勝志
NEC インターネットシステム研究所

概要

企業や組織のネットワークを外部からの不正なアクセス等から守る方法の一つにパケットフィルタリングがある．しかしながら，パケットフィルタリングの設定は多数の複雑なルールから構成されるため，セキュリティ専門家ですら運用管理するのは難しい．

本稿では，パケットフィルタリングのルール集合を詳細に分析することができるマトリックスを用いた分析手法について述べる．マトリックスとは，ルール集合で作る多次元空間を，各ルールの各条件属性の範囲指定された全ての境界点で区割りしてできる最小の多次元立方体である．このマトリックスとルールを対応付けることで，不要なルールを検出することが可能となる．また，実装したシステムを用いて，分析実行速度の評価実験を行った．その結果，中規模程度のルール集合（500 ルール程度）ならば実用的な速度で分析が可能であることが分かった．更に，実際のルール集合を用いて不要なルールの検出を行ったところ，525 ルールの中から 44 個の不要なルールを検出することができた．

A Packet Filtering Rules Analysis by Decomposing into Matrixes
- Removable and Revisable Rules Detection -

Katsushi MATSUDA
Internet Systems Research Laboratories, NEC Corp.

Abstract

Packet filters are essential for organizations that are connected to the Internet. However, it is difficult for even security experts to manage the filters, because their configurations consist of a large number of rules.

In this paper, we describe a novel analysis method using matrixes. The matrixes are the smallest hyper cubes into which a hyper space including a rule set is decomposed at all boundary derived from every beginning and end value of each attribute of the all rule. Mapping these matrixes to the rules enables to find unnecessary rules. We have had an experiment of elapse time of analyzing. The result shows that the system produces results within practical time up to about five hundreds rules. And also we have detected 44 unnecessary rules from an actual rule set consisted of 525 rules.

1 はじめに

企業や組織のネットワークを外部からの不正なアクセス等から守る方法の一つにパケットフィルタリングがある．パケットフィルタリングは，複数のネットワークを接続するゲートウェイやルータに設置されるネットワーク機器またはソフトウェアであり，ネットワークを流れるパケットの属性をルールと照合することで，そのパケットの通過の可否を決定し，内部ネットワークを守る．

パケットフィルタリングにより内部ネットワークを守ることは可能であるが，そのためにはパケットフィルタリングのルール集合を適切に設定･管理する必要がある．しかしながら，パケットフィルタリングのルール数は数 10～数 100，時には数 1000 に及ぶ．しかも各ルールの条件は複数の属性の組

み合わせになるため，ネットワーク管理の専門家ですらルール集合全体としてどうなっているのかを理解することは難しい．

特に，ルール集合の管理においては，時間とともに単調増加するルール数を如何に減少させて管理を容易にするかが問題である．実際，長期間運用しているルール集合には，なくてもルール集合全体の意味が変らないルールが多数存在する．このようなルールは無用にルール数を増加させ，その結果ネットワーク管理者の管理コストを増大させている．

本稿では，パケットフィルタリングのルール集合を詳細に分析することを可能にするマトリックス分解の手法と，それを用いて不要なルールを検出する方法について述べる．

2 関連研究

パケットフィルタリングルールの分析で最も基本的な研究は，packet classification という分野である．これは，あるパケットがどのルールと照合するかを速度とメモリ使用量の点から研究するというものである（文献[1][2]に詳しい）．計算幾何学のデータ構造とアルゴリズムを用いることで速度とメモリ使用量のトレードオフに挑んでいる．

また，packet classification と類似した研究の成果として，関連するルールを抽出するクエリーベースのツールがある[3][4]．これらのツールは非常に強力であるが，適切なクエリーを作成するためには高レベルな訓練が必要である．

本稿で述べる不要なルールを検出する先行研究には，[4][5][6][7]等がある．これらの研究は全て形式的に不要なルールを求めている．特に[4]は制約論理プログラミングを用いて論理的に導出しているところが興味深い．

近年，パケットフィルタリングの分析に関する研究は盛んに行われているが，そのほとんどは高レベルな管理ポリシーをターゲットとしており，ファイアウォール等の低レベルなルールを対象としている研究は少ない[7]．

3. マトリックス分解

パケットフィルタリングルールを分析する場合，一般的には条件範囲の関連性をルール単位で比較することが多い．1 対 1 のルールの比較で判断できる場合はこれでも良いが，多対 1 や多対多のルールの比較でないと判断できない分析の場合は，この方法は困難である．

本稿で報告するルール分析方法は，ルール単位ではなく，マトリックスと呼ぶ極小多次元立方体単位によるものである．マトリックスとは，ルール集合で作る多次元空間を，各ルールの各条件属性の範囲指定された全ての境界点で区割りしてできる最小の多次元立方体である．

ルール集合からマトリックスを作成し，ルールとマトリックスの対応付けを行うことをマトリックス分解と呼ぶ．以下，マトリックス分解について詳細に述べる．

3.1 マトリックス生成

ルール集合からマトリックスを作成するのがマトリックス生成フェーズである．一般的なパケットフィルタリングシステムでは，5 種類以上の条件属性（SrcIP，SrcPort，DestIP，DestPort，Protocol）が用いられているが，ここでは簡単のため，2 種類で説明する．また，それらの条件属性は 0 から始まり，比較的小さな値（例えば，13 や 15 等）で終わるものとする．

図 1 に 8 個のルールを持つルール集合の例とそれを 2 次元平面で表現する．優先順位は R1 が最も高く，R8 がデフォルトルールである．ルールの書式は（*第 1 属性の範囲，第 2 属性の範囲，アクション*）とする．アクションは A が accept（通過を許可），D が deny（通過を禁止）を表し，2 次元平面では A が白，D が灰色となっている．

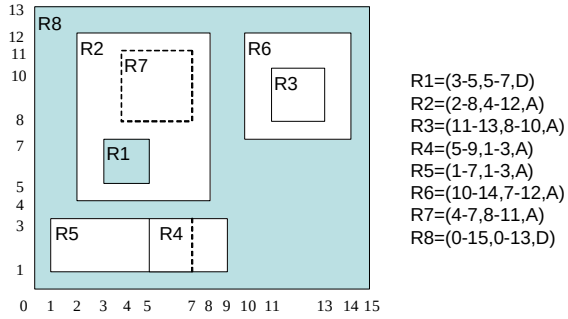

図 1. ルール集合の例

マトリックス生成は，まず条件属性毎にルール集合の各ルールの条件範囲の開始点と終了点（以降，境界点と呼ぶ）を集め，重複をなくした上でソートする．そして各属性を軸として，境界点の交点で極小の多次元立方体を作る．この多次元立方体がマトリックスである．図 1 のルール集合の例では，X 軸（第 1 属性）の境界点が 14 個，Y 軸

(第 2 属性) の境界点が 11 個であるため, 130 個 (=(14-1)*(11-1)) のマトリックスが生成される.

マトリックスは, ルールと類似した書式を持つ. すなわち, (*第 1 属性の範囲, 第 2 属性の範囲, ルールリスト*) である. 以降, この書式のデータをマトリックスデータと呼ぶ.

3.2 ルールとマトリックスの対応付け

マトリックスを生成した後, ルールとマトリックスの対応付けを行う. すなわち, ルールにはそのルールを構成するマトリックスのリストを, マトリックスにはそのマトリックスに関係するルールのリストを列挙することである.

この対応付けによって, ルールの書式は (*第 1 属性の範囲, 第 2 属性の範囲, アクション, マトリックスリスト*) となる. 以降, この書式のデータをルールデータと呼ぶ.

ルールデータのマトリックスリストにはそのリストの順序には意味がないが, マトリックスデータのルールリストの順序はルール集合の優先順位に準拠している必要がある. 例えば, 図 1 の平面図の R1 の部分で説明すると, R1 の下に R2 があり, 更にその下に R8 があるため, R1 を構成するマトリックスデータのルールリストは, (R1,R2,R8) となる.

3.3 実際のデータ構造

3.1 節では, マトリックスの開始点と終了点はそれぞれルール集合で作られる境界点としていたが, 実際のデータ構造では開始点は-0.5, 終了点は +0.5 の値をそれぞれ加えている. ルールに記載される条件範囲の両端の値もその条件に含まれるため, 境界点でマトリックスを作ってしまうと, その境界点を値に持つマトリックスが複数生成され, その境界点上のパケットがどのマトリックスに属するかが判定できないためである. これを図示したものが図 2 である.

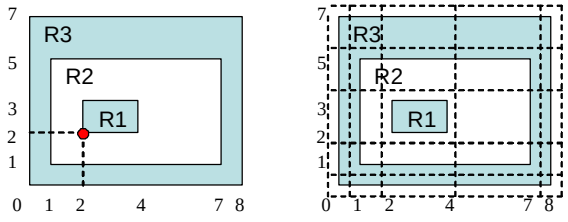

図 2. マトリックスの境界

例えば, 第 1 属性の値が 2, 第 2 属性の値が 2 のパケットを考えると, そのパケットは (1-2,1-2) と (2-4,1-2) と (1-2,2-3) と (2-4,2-3) の 4 個のマトリックスに属してしまう. パケットの属性は離散値であるため, マトリックスの境界を属性値の系列から外すことで, マトリックスの一意性を確保することができる. 図 2 では, 整数値を 0.5 外してマトリックスの境界としている. これによって, 上述のパケットは, (1.5-4.5,1.5-3.5) のマトリックスに一意に属することとなる.

3.4 マトリックスの削除

マトリックス分解によって生成されるマトリックスの数は, 各属性の境界点の積によって定まる. 例えば, ルール集合に 50 個のルールがあり, ルールの条件属性が 5 種類, 各ルールの条件範囲の開始点と終了点に重なりがない場合, 95 億個を超えるマトリックスが生成される. すなわち, ルール数を n, 条件属性数を d とすると, $(2n-1)^d$ 個のマトリックスが生成される.

上述のマトリックス数は原理上の最大値で, 実際にはそのようなルール集合はありえない. 現実に用いられているルール集合を調べたところ, 平均的には $(3n/2)^{d/2}$ 程度である. しかしながら, この計算式であっても, 50 個のルールの場合, 49,000 個になる. マトリックス数は, メモリ使用量に影響する上, 後述のルール分析速度にも関わってくるため, 少しでも減らす必要がある.

マトリックスを削減するために, デフォルトルールにのみ関係しているマトリックスを削除する. すなわち, マトリックスデータのルールリストを参照し, そのリストがデフォルトルールのみの場合は, そのマトリックスデータを削除する. 後述のルール分析において, マトリックスデータが存在しない領域があっても, その領域はデフォルトルールであることが一意に分かるため, このようにマトリックスを削除しても問題はない.

3.5 マトリックス分解の意味

マトリックス分解によって, ルール集合をルールデータとマトリックスデータに変換することは, ルールの条件範囲をこれ以上細分化しても無意味という極小領域で表現することでインデックス化していることになる. また, ルールデータはルールをマトリックスの集合として扱う手段を提供する. これによって, ルール間の関係やルールの状態を集合論的に扱うことができる.

次節では, マトリックスを用いて不要なルールを検出する方法について述べる.

4 ルール分析

マトリックス分解によって様々なルール分析が可能になる. 2 節で述べた packet classification についてもマトリックスを使うことで, メモリ使用量は平均 $O(n^{d/2})$と膨大であるが, 速度は $O(1)$を実現することが可能である. 更に単なるパケットだけではなく, あるパケットの範囲についても同じ速度で実行可能である.

本節では, 時間の経過とともにルール数が増加したルール集合から, 不要なルールおよび冗長な条件を持つルールを検出するルール分析について述べる. このようなルールを検出･削除することで, ルール数を減少させ, ルール集合のメンテナンス性を向上させることが可能となる.

4.1 不要ルールの検出

不要ルールとは, そのルールがルール集合から省かれた場合でも, ルール集合全体の動作に違いがないようなルールである. 一般的には, このような不要ルールは 2 種類ある. 1 つは, どのようなパケットが到達しても発火しないルールであり, 他方は, そのルールがなくても優先順位の低い別のルールが同じアクションを行う場合である. ここでは前者を潜伏ルール, 後者を冗長ルールと呼ぶ.

マトリックスによって, これらの潜伏ルールと冗長ルールを定義することが可能である. すなわち, あるルール R_i が潜伏ルールであることを Concealed(R_i), 冗長ルールであることを Verbose(R_i)とすると, 以下のようになる.

$$\text{Concealed}(R_i) \text{ iff } \quad \forall m\left\{(m \in R_i) \subseteq \sum_{j=1}^{i-1} R_j\right\} \tag{1}$$

$$\text{Verbose}(R_i) \text{ iff } \quad \forall m(m \in R_i)\{act(R_i) = act(next(m, R_i))\} \tag{2}$$

ここでルール集合を $R=\{R_1,R_2,\cdots,R_n\}$, マトリックスを m, ルール R_iのアクションを act(R_i), マトリックス mのルールリスト中のルール R_iの次のルールを next(m,R_i)とする. なお, 式(1)は, $i>1 \wedge i \neq n$, 式(2)は $i \neq n$ である.

式(1)の定義によると, R_i を構成するマトリックス全てについてそのルールリストに R_i より優先順位の高いルールが存在すれば, ルール R_iは潜伏ルールである. 一方, 式(2)の定義によると, R_iを構成するマトリックス全てについてそのルールリストの R_iの次のルールのアクションが R_iと同じアクションであれば, ルール R_iは冗長ルールである.

式(1)および式(2)はあるルールが不要ルールである十分条件であるが, 必要条件ではない. すなわち, (1)と(2)を組み合わせた以下の式(3)が必要十分条件となる.

$$\text{Removable}(R_i) \text{ iff } \quad \forall m(m \in R_i)\left(m \subseteq \sum_{j=1}^{i-1} R_j\right) \vee \left(act(R_i) = act(next(m, R_i))\right) \tag{3}$$

ルール集合から不要ルールを検出するためのアルゴリズムは式(3)から容易に作成できる. また, あるルール R_iが不要ルールかどうかの判定は, 最大 $|R_i|$回のマトリックスデータのルールリスト調査で可能である.

4.2 冗長条件ルールの検出

冗長条件ルールとは, そのルールの条件属性の一部または全てについて範囲を縮小した場合でも, ルール集合全体の動作に違いがないようなルールである. 不要ルールがルールそのものを削除するのに対して, 冗長条件ルールはルールの一部を削除することになる.

冗長条件ルールによって削除される条件部分についても前節の不要ルールと同じく Concealed と Verbose の状態がある. 更に削除する条件部分を除いた残りの部分は, ルールとして記述可能でなければならないため, 残りの部分は多次元立方体になっている必要がある. 冗長な条件部分を削除する代わりにルール数が増えるのは本末転倒であるため, このような条件を設けている. 以下, マトリックスを用いて冗長条件ルールを定義する.

$$\begin{aligned}&\text{Revisable}(R_i) \text{ iff}\\ &\forall m(m \in M)\exists M(M \subset R_i)\exists k \forall l(k,l \in K)\{\\ &range(R_i,k) \neq range(M,k)\\ &\wedge range(R_i,l) = range(M,l)\\ &\wedge (R_i \neq top(m) \vee act(R_i) = act(next(m, R_i)))\}\end{aligned} \tag{4}$$

ここで M は多次元立方体のマトリックス集合, K は条件属性の全集合, M の属性条件 k の範囲を range(M,k), マトリックス m のルールリストの先頭ルールを top(m)とする.

式(4)は 1 番目と 2 番目の論理項で, 1 種類の条件属性を除く他の条件属性が R_iと同じであるマトリックス集合 M を規定している. そしてそのような M の構成マトリックス全てが Concealed か Verbose の

いずれかの場合に R_i が冗長条件ルールとなる.

不良ルールと同様に, ルール集合から冗長条件ルールを検出するためのアルゴリズムは式(4)から容易に作成できる. また, あるルール R_i が冗長条件ルールかどうかの判定は, 属性の種類が k 個の場合, 高々 $2k$ 回のマトリックス集合の生成と調査で可能である.

5 フィルタリングルール分析システム

3 節で述べたマトリックス分解の仕組みを用い, 4 節で述べた不要ルールと冗長条件ルールの検出を行うルール分析システムを開発した. 分析システムは, Microsoft Windows Xp のアプリケーションで, Microsoft Visual C++ 6.0 で開発している. ルール集合の記述は, アクセス制御やファイアウォール設定や侵入検知システム設定を共通のフォーマットで記述できる汎用ポリシー記述言語 SCCML[8]を用いた.

図 3 に用いたルール集合の例を示す. このルール集合は Cisco IOS のパケットフィルタリングの例 (一部省略している) であり, 時間とともにルールが増加し, 追加したルールはルール集合の上部に挿入されていったものとしている. 各ルールの詳細は説明は省くが, サーバが増減したり, サービスが開始終了したり (実際はサービスが交代している), と実際の運用によって生成される設定に近いルール集合となっている.

```
    ! policy
001 deny ip 10.33.109.0 0.0.0.255 10.26.192.0 0.0.0.255
002 permit tcp 10.33.138.0 0.0.0.255 host 10.26.195.172 range 20 23
003 permit tcp 10.33.138.0 0.0.0.255 host 10.26.195.172 eq www
004 permit tcp 10.99.3.0 0.0.0.255 host 10.26.192.3 range 20 21
005 permit ip 10.33.138.0 0.0.0.255 10.26.192.0 0.0.0.255
006 deny ip 10.33.109.0 0.0.0.255 host 10.26.192.10 range 0 19
007 deny ip 10.33.109.0 0.0.0.255 host 10.26.192.10 range 24 65535
008 permit ip 10.33.109.0 0.0.0.255 host 10.26.192.10
009 permit ip 10.33.109.0 0.0.0.255 10.26.192.0 0.0.0.255
010 deny ip 10.26.224.0 0.0.224.255 host 10.26.192.10
011 permit tcp 10.26.0.0 0.0.255.255 host 10.26.192.9 eq ssh
012 permit tcp 10.26.0.0 0.0.255.255 host 10.26.192.9 eq telnet
013 deny ip any host 10.26.192.5
014 permit tcp 10.26.0.0 0.0.255.255 host 10.26.192.10 range 20 21
015 deny tcp any gt 1023 host 10.26.192.192 eq www
016 permit ip 10.0.0.0 0.255.255.255 10.26.192.0 0.0.0.248 lt 1024
017 permit tcp any gt 1023 host 10.26.192.192 eq www
018 deny ip any any
```

図 3. ルール集合の例

SCCML への変換においては, 先頭から順にルール番号を割り当てている. このルール集合を SCCML に変換した後, 分析システムに読み込ませ, 不要ルールと冗長条件ルールの検出を行った結果が図 4 である.

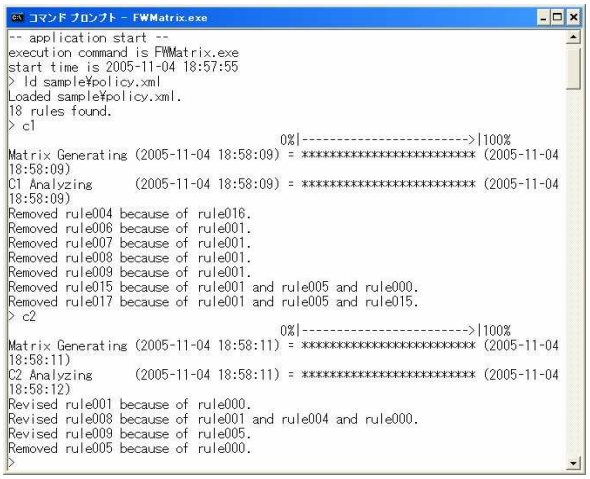

図 4. システムの実行例 (1)

c1 とは, 不要ルールの検出と削除を行うコマンドで, c2 とは, 冗長条件ルールの検出と修正を行うコマンドである. なお, c2 は冗長条件ルールの修正を行った後, 自動的に c1 を実行する. なお, デフォルトルール (図 3 の 018 行目) は, 分析システム上では rule000 と番号付けされる.

c1 の実行後の結果で, "Removed rule004 because of rule016" のように原因のルール (rule016) が不要ルール (rule004) より優先順位が低い場合はその不要ルールは Verbose であり, 逆の場合は Concealed である. 両方の原因ルールがある場合 (rule015) は Verbose と Concealed の組み合わせである.

c2 の実行結果についても c1 と同様である. 特筆すべき点は, 一部のルールの条件部分が縮小されたため, 最初の c1 では検出されなかった rule005 が不要ルールとして検出·削除されたことである.

不要ルールの検出·削除と冗長条件ルールの検出·縮小によって, 図 3 のルール集合の場合, 18 個のルールが 10 個のルールになる. このように本システムを用いることでルール数を削減でき, ネットワーク管理のコストを低減させることができる.

6 性能評価

開発した分析システムの性能評価を行った. ここでの評価指標は, 実行速度と圧縮率である.

6.1 実行速度

マトリックス分解と不要ルール検出·削除の分析の実行速度をルール数毎に計測した. 図 5 に計測結果のグラフを示す.

今回の実験に用いるルール集合を自動的に生成するツールを別途開発した. 3.4 節で述べた平均的に生成されるマトリックス数は$(3n/2)^{d/2}$個程度であるが, このマトリックス数に合わせた生成ツールを開発することは難しい. そのため, 近似的に同程度のマトリックス数となるように各条件属性の生成確率を調整し, 生成するマトリックス数が $n^{2.57}$ 個程度になるように設計した. dは一般的に5であるため, 平均的なマトリックス数は$(3n/2)^{2.5}$ となる. nが比較的小さい値の場合, $n^{2.57}$と近い値となるため, 実験には適当である.

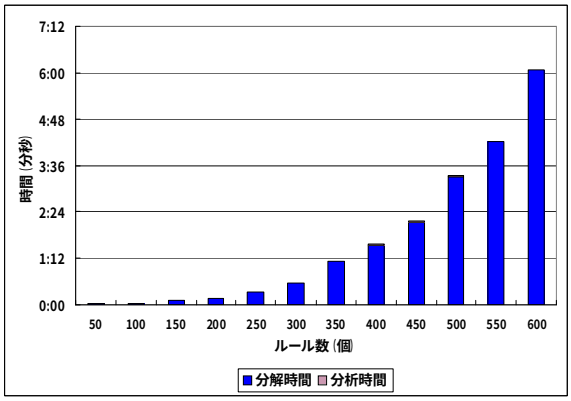

図 5. 実行速度の計測結果

グラフでは分かりにくいが, ルール数 250 個までは分析時間が0秒で, それ以上は1秒であった. この結果, 500 ルール前後の中規模なルール集合までならば十分実用の範囲で分析ができることが分かった.

また, 3.4 節で述べたデフォルトルールにのみ関係するマトリックスを削除する方法についても測定した. 実行速度を測定したルール集合の場合, ルール数の多少とは関係なく, 平均 74.8%のマトリックスが削除された.

6.2 圧縮率

圧縮率とは, 不要ルールの検出･削除の結果, ルール集合のルール数が元のルール数の何%になるか, という指標である. 圧縮率の測定には 6.1 節で用いたルール自動生成ツールは用いることができない. そのため, 筆者の所属している部門のルータのフィルタリングルールを用いて測定を行った.

実験に用いたルール集合は, Cisco IOS のフィルタリングルールであり, 525 個のルールがあった. IOS の記述から SCCML に変換し, システムで不要ルールの検出を行った. その結果, 44 個のルールを削除することができた. 圧縮率は 91%である.

7 おわりに

パケットフィルタリングのルール集合をマトリックスを用いて表現することで不要ルールや冗長条件ルールが検出できることを示した. また, 実装した分析システムで評価実験を行い, 500 ルール程度の中規模パケットフィルタリングシステムでも十分実用的な速度で分析ができることを示した. また, 実際に運用されているルール集合を分析し, 525 個のルールから 44 個のルールが不要なルールであることを検出した.

本稿では, 不要ルールと冗長条件ルールの検出について報告したが, マトリックスを用いることで更に様々な分析が可能になるものと考える. 他の分析についても引き続き検討する予定である.

参考文献

[1] Feldmann and muthukrishnan, “Tradeoffs for Packet Classification”, Proceedings of INFOCOM(3), pp.1193-1202, 2000.

[2] Gupta and McKeown, “Algorithms for Packet Classification”, IEEE Network, Vol.15, No.2, pp.24-32, 2001.

[3] Mayer et al, “Fang: A Firewall Analysis Engine”, Proceedings of 2000 IEEE Symposium on Security and Privacy, pp.177-187, 2000.

[4] Eronen and Zitting, “An Expert System for Analyzing Firewall Rules”, Proceedings of 6th Nordic Workshop on Secure IT-Systems (NordSec2001), pp.100-107, 2001.

[5] Jalili and Rezvani, “Specification and Verification of Security Policies in Firewalls”, EuroAsia-ICT 2002, LNCS 2510, pp.154-163, 2002.

[6] Kurland, “Firewall Builder”, 11th DFN-CERT Workshop, ISBN3-00-012959-6, 2004.

[7] Al-Shaer and Hamed, “Modeling and Management of Firewall Policies”, IEEE Transaction on Network and Service Management, Vol.1-1, 2004.

[8] 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 Vol.2004, No.129, pp.89-94, 2004.