2 SELinuxのアーキテクチャとアクセス制御モデル

海外浩平（日本電気(株) OSS 推進センター）

SELinux（Security Enhanced Linux）は，LSMに対応した強制アクセス制御機構の1つで，元々は米国国家安全保障局（NSA；National Security Agency）の研究プロジェクトから生まれたFLASK（Flexible advanced security kernel）アーキテクチャをLinux上に移植したものである．この"Flexible"の意味するところは，従前の強制アクセス制御がTCSEC（Trusted Computer System Evaluation Criteria）の要求するような機密階層・機密区分を前提としていたことに対し，より柔軟なセキュリティポリシー記述を可能にしたということである．

最初のバージョンのSELinuxはv2.4系カーネルのパッチとして2000年12月に一般に公開され，その後，v2.6系Linuxカーネルの開発過程である2003年8月にメインライン化されている．現在はNSAのほかにRed Hat，Tresys，IBM，HP，NECをはじめ多くの企業・開発者からなるコミュニティによって開発が進められており，Red Hat Enterprise LinuxやFedora Linuxをはじめとする多くのLinuxディストリビューションが対応している．

本稿では，SELinuxのアーキテクチャと，そのアクセス制御モデルを解説する．

アーキテクチャ

SELinuxはクライアント・サーバモデルに類似したアーキテクチャを持つ．図-1に示すように，セキュリティサーバであるSELinuxは，オブジェクト・マネージャの問合せに対して，自身の保持するセキュリティポリシーに基づいて意思決定を行い，それを「許可」すべきか，「禁止」すべきかを回答する．問合せを行ったオブジェクト・マネージャは，セキュリティサーバの意思決定に従って，利用者から送られた要求を実行するか否かを決定する．

オブジェクト・マネージャとは，ファイルやネットワーク等の計算機資源（オブジェクト）を管理するソフトウェアモジュールで，たとえばファイルへの書き込みなど，利用者から要求された操作を自身の管理下にあるオブジェクトに対して実行する．その際に，利用者が要求した操作を実行する権限を有しているかをチェックするアクセス制御を併せて実行する．典型的には，OSのファイルシステムやネットワークシステムは，ファイルやネットワーク資源を管理するオブジェクト・マネージャである．

たとえばファイルにデータを書き込むためにwrite(2)システムコールを呼び出す，ソケットを介してパケットを送出するためにsend(2)システムコールを呼び出す，といった行為をもう少し一般化してみよう．

利用者（および，その代理人であるプロセス）は，ファイルシステムの管理下にあるファイルやディレクトリを，あるいはネットワークシステムの管理下にあるソケットを直接利用することはできない．これらのオブジェクトを利用するためには，対象となるオブジェクトを管理するオブジェクト・マネージャにリクエストを送出し，利用者の必要とする操作を代わりに行ってもらう必要がある．ファイルシステムやネットワークシステムはOSカーネルの一部

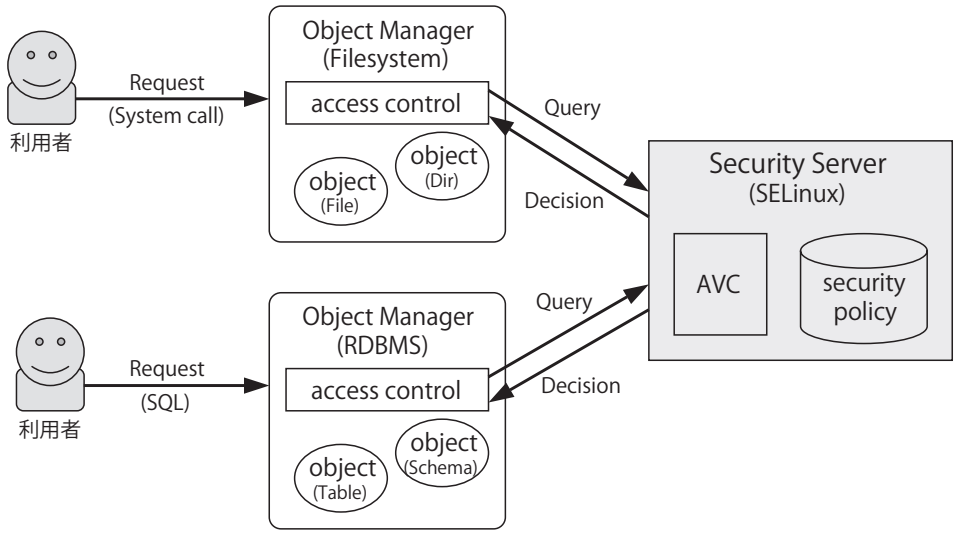

図-1　セキュリティサーバとオブジェクト・マネージャ

であるので，これらのリクエストはシステムコールという形で発行される．

オブジェクト・マネージャは，たとえばファイルへの書き込みなど，要求された操作を自身の管理下にあるオブジェクトに対して実行する．だがその前に，オブジェクト・マネージャは利用者が要求した操作を実行して構わないのか否か，アクセス制御を行う必要がある．

その際に，オブジェクト・マネージャはセキュリティサーバである SELinux に問合せを行い，利用者から要求のあった操作を実行すべきか否か，SELinux の意思決定を要求する．SELinux は内部にセキュリティポリシーと呼ばれる巨大なルール集を保持しており，セキュリティポリシーに照らし合わせて，オブジェクト・マネージャからの問合せを許可すべきか禁止すべきかという意思決定を行い，これを呼び出し元に回答する．なお，この意思決定はやや重い処理であるため，SELinux は直近の意思決定の結果を AVC（access vector cache）と呼ばれる領域にキャッシュしておき，オブジェクト・マネージャからの問合せに対して迅速に回答できるよう設計されている．

『許可』または『禁止』という，SELinux の意思決定を受け取ったオブジェクト・マネージャは，その結果に従って利用者からのリクエストをそのまま実行するか，実行を直ちに中断しエラーを返すかのどちらかの動作を行う．

なお，オブジェクト・マネージャは必ずしも OS の一部というわけではないことに留意されたい．X Window System は Window や Font といった X オブジェクトを管理し，利用者は X プロトコルを介してこれらを操作する．RDBMS はテーブルやスキーマといった DB オブジェクトを管理し，利用者は SQL を介してこれらを操作する．オブジェクト・マネージャが OS カーネルであるか否か，リクエスト発行形態がシステムコールであるか否か，細部で異なる点はあるが，本質的に大きな違いはない．

ファイルシステムが UNIX パーミッション機構を持っているように，個々のオブジェクト・マネージャが独自のアクセス制御を行う場合もある．しかし，この仕組みのポイントは，個々のアクセス制御とは独立に，セキュリティサーバ（SELinux）が集中管理された自身のセキュリティポリシーに基づいて一元的に意思決定を行うという点である．たとえば，共有のオブジェクトに何か機密情報を書き出して他の利用者に開示することができるという点では，ファイルもデータベースも大差はない．だが，アクセス制御の結果，一方は禁止，一方は許可というのでは，情報漏えいを防ぐという観点からは都合が悪い．SELinux はすべての意思決定を一元化することで，どのオブジェクト・マネージャを利用するかにかかわらず，首尾一貫したアクセス制御を行うことができる．

```
[kaigai@saba ~]$ ls -Z /var/log/messages*
-rw-r-----. root adm system_u:object_r:var_log_t:s0  /var/log/messages
-rw-r-----. root adm system_u:object_r:var_log_t:s0  /var/log/messages-20100530
-rw-r-----. root adm system_u:object_r:var_log_t:s0  /var/log/messages-20100606
-rw-r-----. root adm system_u:object_r:var_log_t:s0  /var/log/messages-20100613
-rw-r-----. root adm system_u:object_r:var_log_t:s0  /var/log/messages-20100620

[ymj@saba ~]$ ps -Z
LABEL                             PID TTY          TIME CMD
staff_u:staff_r:staff_t:s0       8816 pts/5    00:00:00 bash
staff_u:staff_r:staff_t:s0       8870 pts/5    00:00:00 ps
```

図 -2　セキュリティコンテキスト

セキュリティコンテキスト

オブジェクト・マネージャが SELinux に問合せを行う際に，対象であるオブジェクトを識別する方法を，クライアント・サーバシステムとの比較で考えることにする．たとえば我々が Web ページを参照するとき，クライアント側は Web サーバに URL を送出することで参照したいページをサーバ側に伝達する．サーバ側は渡された URL を解釈することで，要求された Web ページを正確に識別することができる．一方，SELinux では，オブジェクト・マネージャがセキュリティサーバ（SELinux）に問合せを行う際に，誰の何に対するアクセスについて意思決定を求めているのかをセキュリティサーバ側が正確に識別できるよう，オブジェクト・マネージャ側から伝達する必要がある．だが，Web サーバと異なり，SELinux は広範な種類のオブジェクトを取り扱う必要があるため，あらゆる種類のオブジェクトに共通でしかもアクセス制御の識別子として適切な属性を抽出するのは困難である．たとえば，Web サーバにおける HTML 文書のように，すべてのオブジェクトがファイルシステム上に存在すればパス名は識別子の候補となり得るが，SELinux はそのほかにも，ネットワークソケット，共有メモリ等プロセス間通信オブジェクトや，X Window System や RDBMS などの管理下にあるユーザ空間オブジェクトもその対象としている．したがって SELinux では，パス名，所有者 ID，パーミッションなど，オブジェクトがすでに持っている何らかの属性情報をアクセス制御に用いるのではなく，セキュリティコンテキストと呼ばれる SELinux 専用の新たな属性をオブジェクトに付加して，これをアクセス制御の意思決定を行う際の識別子として用いる．

図 -2 は，SELinux がインストールされたマシン（Fedora rawhide）上で，ファイルの情報を表示する ls コマンド，およびプロセスの情報を表示する ps コマンドを実行した結果である．各コマンドを実行すると，太字で示すようにコロン（:）区切りの独特な文字列が表示される．この文字列をセキュリティコンテキストと呼ぶ．SELinux はオブジェクトの識別にセキュリティコンテキストのみを用いるため，所有者やファイル名など，その他のオブジェクトの属性はアクセス制御に影響を与えない．

オブジェクト・マネージャは，SELinux に対して“誰が（アクセス主体；Subject）”，“何に（アクセス対象；Object）”，“何をできるか（Action）”の組を問い合わせ，SELinux は要求されたアクセスの可否を回答する．このとき，アクセスの主体である Subject とその対象となる Object を識別するためにセキュリティコンテキストを用いる．

たとえば，Subject であるセキュリティコンテキスト "staff_u:staff_r:staff_t:s0" を持つプロセス（例：cat コマンド）が，Object であるセキュリティコンテキスト "system_u:object_r:var_log_t:s0" を持つファイル（例：/var/log/messages）を参照するために open(2) を呼ぶとき，このシステムコールを処理するのは OS のファイルシステムである．ファイルシステムは，SELinux がこのファイルへのアクセスを許可するか否か確認するため，プロセスとファイルのセキュリティコンテキストのペアを引数として SELinux

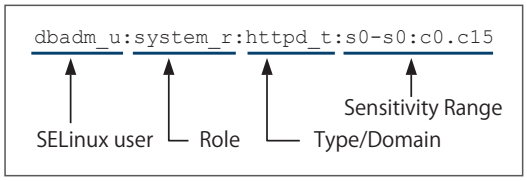

図-3　セキュリティコンテキスト書式

に問合せを行い，SELinux はその情報に基づいてセキュリティポリシーを探索，アクセス制御の意思決定を行う．

なお，図-3 のセキュリティコンテキスト書式に示すように，セキュリティコンテキストは区切り文字によって 4 つのフィールドに分割される．各フィールドにはそれぞれ固有の意味があり，次節以降で説明する SELinux のアクセス制御モデルにおいて，Type/Domain，Role，Sensitivity Range および SELinux user の順に，これがどのように利用されるかを解説する．3 番目のフィールドは，プロセスに付与されたセキュリティコンテキストの場合に限り，ドメインと称する．それ以外の場合はタイプと呼ぶが，両者の間に名前以外の本質的な違いはない．また，伝統的な強制アクセス制御機構では，セキュリティコンテキストの代わりにラベルという呼び方をする．両者に本質的な違いはないが，セキュリティコンテキストの 4 番目のフィールド（Sensitivity range）を狭義にラベルと呼ぶ場合もあるので注意が必要である．また，SELinux の標準セキュリティポリシーでは，慣習的に，SELinux user に "_u"，Role に "_r"，Type/Domain に "_t" の接尾辞をつけ，各シンボルが何を表しているのか一目で分かるようになっている．

■セキュリティコンテキストの関連付け

セキュリティコンテキストもオブジェクトの属性の 1 つであり，所有者 ID やパーミッションと同様に，正しくオブジェクトに関連付けるのはオブジェクト・マネージャの役割である．したがって，ファイルシステムは各ファイルにセキュリティコンテキストを関連付ける機能を有している必要があり，ネットワークシステム，データベースシステム等でも同様である．

SELinux は，個々のファイルにセキュリティコンテキストを関連付けるために XATTR（eXtended ATTRibtue ; 拡張属性）を用いており，setxattr(2) や getxattr(2) システムコールを用いてアクセスできる．最近の Linux では主要なファイルシステムの大半が XATTR 機能に対応している．

プロセスやソケットなどのインメモリオブジェクトは，これらを内部的に表現する構造体に，セキュリティコンテキストを保持するためのメンバが追加されている．たとえば，プロセスの場合は task_strunt 構造体の cred->security メンバ（void * 型）にセキュリティコンテキストを保持する．

■デフォルトセキュリティコンテキスト

あるオブジェクトを新規に作成するとき，オブジェクト・マネージャは SELinux に問い合わせ，新しいオブジェクトに関連付けるべきセキュリティコンテキストを得る．これをデフォルトセキュリティコンテキストと呼び，誰の作成した何のオブジェクトに，どのようなデフォルトセキュリティコンテキストを関連付けるかは，セキュリティポリシーによって定義されている．

基本的には，プロセスが fork() したときには親プロセスのセキュリティコンテキストを引き継ぎ，ファイルを作成した際には親ディレクトリのセキュリティコンテキストを引き継ぐ．このように，オブジェクト間の親子関係に基づいてデフォルト値が決まる．だが，いくつかの重要なポイントでは，タイプ／ドメイン遷移という特殊なルールによって例外的なラベル付けを定義することが可能で，たとえば，タイプ遷移を用いて /tmp ディレクトリに生成したファイルに対し，利用者に応じたセキュリティコンテキストを付与して一時ファイルを通じた情報の漏えいを防ぐようになっている．ドメイン遷移については，SELinux における利用者の権限と密接に繋がっているため，より詳しく「ドメイン遷移と RBAC」で説明する．

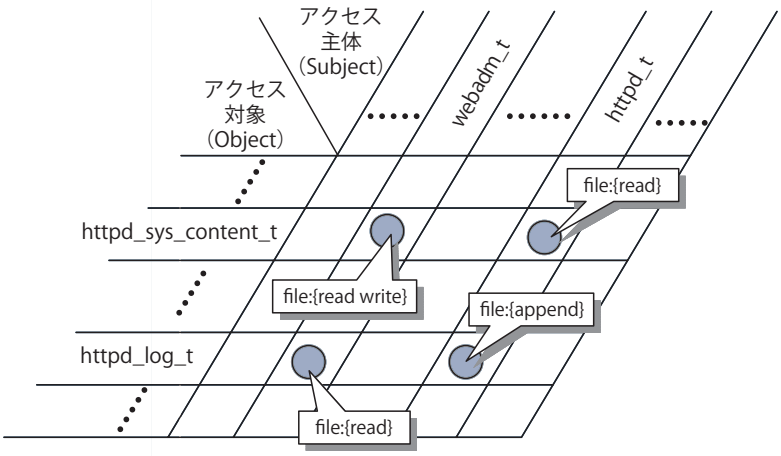

図 -4　アクセス制御マトリックス

SELinux のアクセス制御モデル

SELinux は TE（Type Enforcement），RBAC（Role Based Access Control），MLS（Multi Level Security）の 3 つのアクセス制御モデルを実装している．セキュリティコンテキストの 4 つのフィールドのうち，UNIX ユーザとの関連付けに利用される先頭の SELinux user フィールドを除く 3 つのフィールドが，各々のアクセス制御モデルに対応する．

オブジェクト・マネージャから問合せのあった操作を実行してよいか否かを判断するために，SELinux は TE，RBAC，MLS それぞれのルールに基づいて検査を行い，3 つのアクセス制御モデルのどれか 1 つでも『禁止』の意思決定を行った場合，呼び出し元には『禁止』と回答する．

■ TE（Type Enforcement）

TE（Type Enforcement）は SELinux における中核的なアクセス制御モデルで，セキュリティポリシーの大半は TE 向けのルールである．このアクセス制御モデルではセキュリティコンテキストの 3 番目のフィールド（Type/Domain）を用いる．

図 -4 に示すように，TE のアクセス制御モデルは，巨大なアクセス制御マトリックスとして捉えることができる．SELinux がある操作を許可するか否かを判断するために，その操作を実行しようとするアクセス主体（Subject）のタイプと，その操作によるアクセス対象（Object）のタイプの組合せに対して，アクセス制御マトリックス上では実行可能な操作の組（図 -4 ではフキダシ内に表記）が定義されており，それが要求された操作をすべて含んでいるか否かをチェックする．

前述の通り，セキュリティコンテキストの Type/Domain のフィールドは，それがプロセスに関連付けられているときにはドメインと呼ばれ，それ以外のオブジェクトに関連付けられているときにはタイプと呼ばれる．一般的に何かの操作の実行主体となれるのはプロセスである．プロセスのタイプに対して，オブジェクトと実行可能な操作の組が紐付いていると考えれば，それは権限のセットと等価であるため，特別にドメインと呼ぶことにしている．

ここで，セキュリティポリシーの具体例を図 -5 に示し，TE のアクセス制御ルールがどのように働くのかを説明する．

ルール（1）は，httpd_t ドメインを持つプロセスが，httpd_sys_content_t タイプを持つファイルに対して read 操作を行うことを許可する．write 操作は許可されていない．一方，ルール（2）は，webadm_t ドメインを持つプロセスが，httpd_sys_content_t タイプを持つファイルに対して read および write 操作を行うことを許可する．各タイプ／ドメインの名称は単なる識別子でそれ自身に意味はないが，httpd_t が Web サーバに，webadm_t が Web 管理者のシェルに，httpd_sys_content_t が HTML 文書へと正しく関連付けされていれば，このポリシーは Web 管理者の HTML 文書の編集を許可するものの，Web サーバは HTML 文書を改ざんできないことを保証する．

ルール（3）は httpd_t ドメインを持つプロセスが，httpd_log_t タイプを持つファイルに対して append（追記）操作を行うことを許可し，ルール（4）は webadm_t ドメインを持つプロセスが，httpd_log_t タイプを持つファイルに対して read 操作を行うことを許可する．このポリシーは Web サーバにログ

```
allow httpd_t httpd_sys_content_t : file { read };          ... (1)
allow webadm_t httpd_sys_content_t : file { read write };   ... (2)

allow httpd_t httpd_log_t : file { append };                ... (3)
allow webadm_t httpd_log_t : file { read };                 ... (4)
```

図-5　セキュリティポリシー例(1)

の追記を許可し，Web管理者がそれをチェックすることを許可するが，誰にも書き換え（write）を許していないことが分かる．

伝統的な強制アクセス制御メカニズムでは，機密レベル・機密カテゴリによる利用者データの分離に主眼を置いていた(「MLS」の節を参照のこと)．そのため，少ないルールで効率的にポリシーを記述できるというメリットの反面，たとえば上記のWebサーバ／Web管理者の例に挙げたような，プログラムの特性に応じたセキュリティポリシーを柔軟に記述するといった用途には不向きであった．

これに対して，TEのモデルでは個々のプロセス／プログラムに関連付けられるべきドメインと，操作対象であるオブジェクトの間に許可する操作を逐一記述できるため，セキュリティポリシーの柔軟性が高いと言える．SELinuxの元になったセキュリティアーキテクチャをFLASK（FLexible Advanced Security Kernel）と呼ぶが，"flexible" の意味するところは，伝統的な強制アクセス制御メカニズムとは一線を画すセキュリティポリシー記述の柔軟性である．

一方で，TEのモデルでは個々のドメイン／タイプ間の関係を逐一記述する必要があるため，セキュリティポリシーのサイズが大きくなりがちという傾向がある．なお，ほとんどの場合はコミュニティで開発されている標準のセキュリティポリシーを，Red Hatはじめディストリビュータが自システム向けにカスタマイズして配布しているので，基本的に，利用者が自身でポリシーを記述する必要はない．

■ドメイン遷移とRBAC（Role Based Access Control）

あるプロセスが子プロセスを生成して別のプログラムを実行するとき，基本的には親プロセスのセキュリティコンテキストを引き継ぐ．そのため，たとえばログイン時にstaff_tというドメインが割り当てられたログインシェルは，その後，子プロセスとしてviやlessを起動したとしても，セキュリティコンテキストは不変である．SELinuxの意思決定はセキュリティコンテキストだけに基づいて行われるため，プロセスのセキュリティコンテキストが子プロセスに引き継がれる以上，アクセス制御の結果も変わらない．

だが，Linuxではすべてのプロセスは PID=1 の /sbin/init を祖先に持つ．仮にすべてのプロセスが単純に親プロセスのセキュリティコンテキストを継承したのでは，結局，すべてのプロセスがまったく同一のドメインに属することになり，意味のあるアクセス制御ができなくなる．

実はTEにはドメイン遷移という仕組みが用意されており，LinuxがSet-UIDプログラムの実行時にユーザIDを切り替えるのと似た仕組みで，execve(2)システムコールの実行時にプロセスのセキュリティコンテキストを切り替えることができる．

図-6のセキュリティポリシーは，ドメイン遷移の設定例である．initrc_tドメインを持つプロセス（通常はシステム初期化スクリプト）が，httpd_exec_tドメインを持つファイル(通常は /usr/sbin/httpd)を実行した際に，プロセスがhttpd_tドメイン（通常はWebサーバプロセス)に遷移するよう設定されている．

ルール（1）はinitrc_tドメインを持つプロセスがhttpd_exec_tタイプを持つファイルの実行を許可する．これにより，プロセスはWebサーバを起動できるようになる．ルール(2)はinitrc_tドメインを持つプロセスがhttpd_tプロセスへとドメイン遷移することを許可する．加えて，ルール(3)によってinitrc_tドメインを持つプロセスが，httpd_exec_tタイプを

```
allow initrc_t httpd_exec_t : file { getattr open read execute }; ... (1)
allow initrc_t httpd_t : process { transition };                   ... (2)
type_transition initrc_t httpd_exec_t:process httpd_t;             ... (3)
```

図 -6　セキュリティポリシー例 (2)

持つファイルを実行したときに，プロセスのドメインが httpd_t に遷移するというイベントが発生する．

なお，SELinux には明示的にドメイン遷移を指定するオプションがあり，次の execve(2) の実行で遷移すべきドメインを指定することができる．利用者がシステムにログインする際には，sshd や logind が pam_selinux.so を用いて利用者のセキュリティコンテキストを設定し，次にシェルを起動する際にドメイン遷移が発生する．この場合，ルール (3) は不要だが，ルール (2) に相当する process {transition} 権限は許可しなければならない．

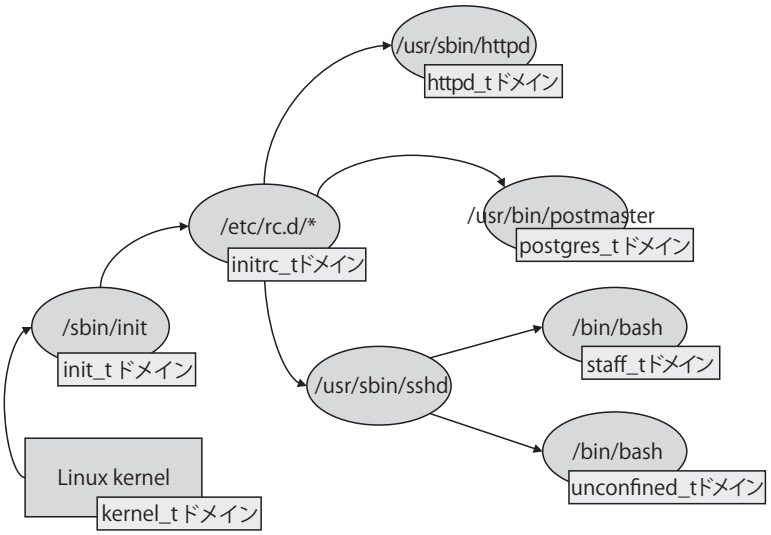

図 -7　ドメイン遷移

図 -7 は，Linux kernel による /sbin/init に始まるシステムの起動から，矢印の順に楕円で示した各プログラム (プロセス) が起動されてゆき，それに伴うドメイン遷移を繰り返しながら各デーモンが固有のドメインで起動されるまでの状態遷移を示したものである．

Linux における Set-UID の場合，Set-UID プログラムを実行した際には例外なくプロセスのユーザ ID が切り替わるが，RBAC を用いることで，SELinux ではドメイン遷移に一定の制約を加えることができる．

セキュリティポリシーによってロールは任意の数のドメインと関連付けられており，プロセスがドメイン遷移を引き起こすときには，遷移元／遷移先が共に現在のロールに関連付けられている必要がある．つまり，ロールによって一定の範囲に制約された中でだけ，ドメイン遷移を行うことができる．

図 -8 に RBAC（Role Based Access Control）の例を示す．ロールは破線で囲んだ範囲であり，0 個以上のドメインを含む．ドメイン遷移の前後でプロセスのドメインはロールで囲まれた範囲を越えられない．たとえば，利用者が dbadm_r ロールに属するとき，DB サーバにアクセス可能な dbadm_t ドメインへの遷移は可能だが，Web サーバにアクセス可能な webadm_t ドメインに遷移することは不可能である．

このチェックは TE のドメイン遷移ルールのチェックとは独立に行われ，どちらか片方のチェックが「禁止」の意思決定を行えば，最終的に，SELinux はプログラムの実行を失敗させる．

なお，ドメイン遷移とは関係ない，プロセス以外のオブジェクトに対しては，セキュリティコンテキストの 2 番目のフィールドに便宜上 "object_r" が割り当てられているが，これはダミーの値であり意味を持たない．

RBAC を用いると TE とは別にドメイン遷移可能な範囲を制約することが可能となり，特定の領域に限定した管理者権限を実装することができる．たとえば，管理者ユーザであっても，サーバの設定ファ

イルの編集や再起動といった管理作業を行う必要がないときには，これらの権限は不要である．したがって，これらの権限を有するドメインへの遷移は，管理作業を行う場合に限っておいた方が，不必要なリスクに曝されないという点で都合がよい．図-8の例では，一般利用者であるstaff_tドメインはWebサーバにもDBサーバにもアクセスできない．Webサーバの管理作業を行う際にはwebadm_tドメインに，DBサーバの管理作業を行う際にはdbadm_tドメインに遷移する必要がある．ログイン時に利用者がdbadm_rロールを選択していれば，staff_tとdbadm_tドメインの両方を含んでいるため，利用者はstaff_tからdbadm_tドメインに遷移できる．だが，Web管理者向けであるwebadm_rロールや，管理作業を行わないstaff_rロールを選択した場合には，dbadm_tドメインはロールの範囲外であるため，ドメイン遷移を行うことができない．つまり，利用者の現在の役割が「DB管理者」ではない場合には，DB管理を行うドメインへの遷移を禁止するということである．

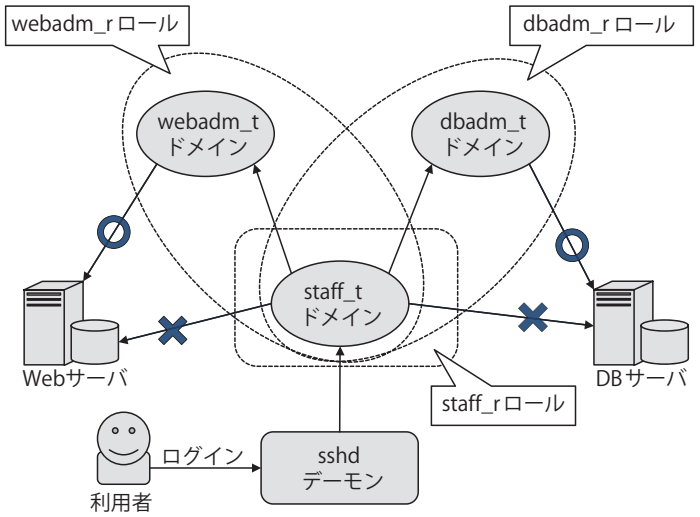

図-8　RBACの例

■ MLS（Multi Level Security）

MLS（Multi Level Security）は，伝統的な機密レベル・機密カテゴリに基づくアクセス制御を実施する．このモデルは元々，1980年代に制定された米国国防総省のセキュリティ評価基準であるTCSEC（Trusted Computer System Evaluation Criteria）から，IT製品のセキュリティ機能評価基準であるISO/IEC 15408（CC；Common Criteria）に受け継がれ，CCのOS向け要件セットの1つLSPP（Label Based Protection Profile）で強制アクセス制御機能として定義されている．

2004～2007年にかけて，SELinux開発コミュニティでは，Red Hat・IBM・HPが中心となりRed Hat Enterprise Linux 5をベースにLSPPの認証を取得するための一連の機能強化が行われた．その最重要項目が（SELinuxが元々有していたTE/RBACのアクセス制御モデルに加えて）MLSを実装することであり，段階的な機能拡張を経てLinuxカーネルv2.6.13で統合された．

MLSはデータの移動する方向に注目したモデルである．read操作をオブジェクト→プロセス方向へのデータの移動，write操作をプロセス→オブジェクト方向へのデータの移動として捉え，ファイルなどの共有オブジェクトを介して機密レベルの高い情報が機密レベルの低いプロセスに，あるいは機密カテゴリを越えた漏えいを防ぐことを目的としている．

MLSの模式図を図-9に示す．機密レベルとは上下関係を定義することが可能な属性で，図の例では，機密レベルの低い順にs0とs1の2つの機密レベルを定義している．機密カテゴリとは包含関係を定義することが可能な属性で，図の例では，c0，c1を要素とする集合，すなわち{c0}，{c1}，{c0,c1}および{}の4種類の機密カテゴリを定義している☆1．

MLSのルールは，read系操作・write系操作の際に，プロセス・オブジェクトが以下の制約条件を満足するかどうかをチェックする．

- read系操作 … プロセスの機密レベル ≧ オブジェクトの機密レベル

☆1 実際の標準セキュリティポリシーにおいては，16段階の機密レベル，1024個の元からなる機密カテゴリを利用することが可能である．

プロセスの機密カテゴリ ⊇ オブジェクトの機密カテゴリ

- write 系操作 … プロセスの機密レベル ＝ オブジェクトの機密レベル
  プロセスの機密カテゴリ ≡ オブジェクトの機密カテゴリ

図 -9 を参照すると，機密レベル s1・機密カテゴリ {c0} を持つプロセス P1 は，同じ機密レベル・機密カテゴリに属するオブジェクト F1 への読み書きが可能である．だが，機密カテゴリ {c1} を持つオブジェクト F2 には包含関係が成立しないため，アクセス不可である．また，P1 は機密レベル s0・機密カテゴリ {c0} に属するオブジェクト F3 に対しては，write 系操作の制約条件を満たさないため，read 系操作のみが可能である．同様に，機密レベル s0・機密カテゴリ {c0, c1} を持つプロセス P2 は，F3 に対して read 系操作のみが可能である．

もう少し口語的な解釈を加えるならば，s1 という高い機密レベルの情報を参照できる P1 は，読み出した情報をより低い機密レベルに漏えいさせることができない．{c0} と {c1} の両方のカテゴリの情報を参照できる P2 は，{c1} から読み出した情報を {c0} に，あるいはその逆に，カテゴリを越えて情報を移動させることができない．つまり，これらの制約条件によって，ある一定の機密レベル・機密カテゴリを持ったデータを一定の範囲内に閉じ込めておくことが可能となる☆2．

MLS ではセキュリティコンテキストの 4 番目のフィールド (Sensitivity range) を用いる．これは上下関係を示す機密レベルと，包含関係を示す機密カテゴリの組合せである．下に例を示す．

```
s0:c1          ... (1)
s2:c0,c4.c6    ... (2)
s0             ... (3)
```

Sensitivity range は，さらに ':' を区切り文字として左右に分割される．(1) を例とすると，"s0" が機密レベル，"c1" が機密カテゴリである．機密カテゴリは集合であるので，複数の要素を持つことがある．(2) の例では "s2:c0, c4.c6" が機密カテゴリだが，区切り文字 '.' は連続した複数の要素を意味するため，結局，この書式は "c0, c4, c5, c6" という 4 つの要素からなる機密カテゴリに属していることを意味する．逆に，空集合である機密カテゴリを持つ Sensitivity range の記法もあり，(3) の例のように機密レベル "s0" だけを持つ場合もある．

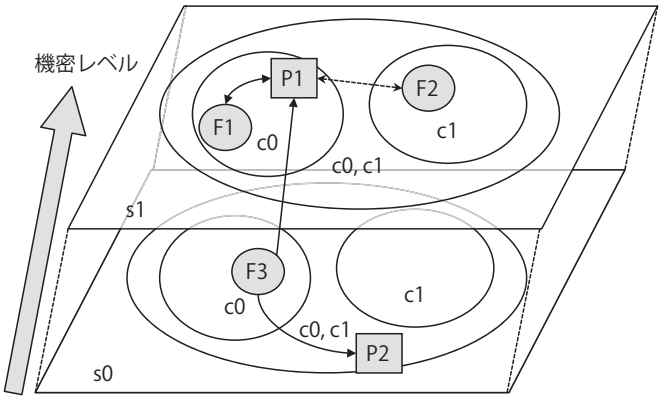

図 -9　MLS 模式図

なお，Red Hat Enterprise Linux や Fedora の標準セキュリティポリシーでは，MLS を簡略化した MCS (Multi Category Security) と呼ばれるルールが搭載されている．これは，上下関係を規定する機密レベルを "s0" の 1 つだけしか定義しないことにより，事実上，機密カテゴリの包含関係によってのみアクセス権のチェックを行うものである．また，制約式も以下のように修正されており，Write 系操作が機密カテゴリの厳密な一致ではなく，包含関係に変更されている．したがって，MCS では，プロセスの機密カテゴリが複数の機密カテゴリを包含するとき，それらの間のデータ移動を防ぐことはできない．

- read 系操作 … プロセスの機密カテゴリ ⊇ オブジ

☆2 機密レベルの低いプロセスに機密レベルの高いファイルへの write 系操作を認めるモデルもあるが，SELinux では同一の機密レベル・機密カテゴリに限って write 系操作を許可している．

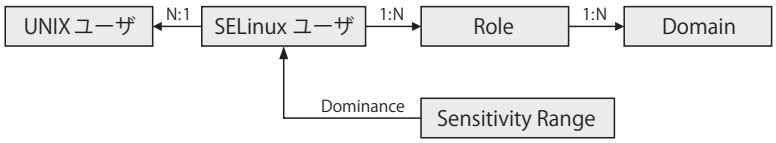

図-10　各フィールドの対応関係

```
[kaigai@saba ~]$ ssh tak@localhost
tak@localhost's password:
Last login: Wed Jun 23 15:09:46 2010 from localhost
[tak@saba ~]$ id -Z
tak_u:user_r:user_t:s0

[kaigai@saba ~]$ ssh tak/staff_r@localhost
tak/staff_r@localhost's password:
Last login: Wed Jun 23 15:10:01 2010 from localhost
[tak@saba ~]$ id -Z
tak_u:staff_r:staff_t:s0
```

図-11　ssh で明示的に Role を指定

ェクトの機密カテゴリ

- write 系操作 … プロセスの機密カテゴリ ⊇ オブジェクトの機密カテゴリ

認証とセキュリティコンテキスト

ここまで TE・RBAC・MLS の説明の中で言及していなかった，セキュリティコンテキストの 1 番目のフィールド（SELinux user）は，UNIX ユーザとの関連付けのために利用されるグループ化のための機構である．

利用者がシステムにログインする際，sshd や logind など認証デーモンは（ユーザ名とパスワードの入力など）何らかの方法で認証を行い，成功すれば，ユーザ ID を変更してログインシェルを起動する．このとき，認証デーモンはユーザ ID だけではなく，利用者に応じたセキュリティコンテキストも変更してから，ログインシェルを起動している．

図-10 に，セキュリティコンテキストの各フィールドと UNIX ユーザの関係を示す．図-10 の通り，SELinux ユーザには複数の UNIX ユーザを関連付けることができる．たとえば，alice，bob という UNIX ユーザが staff_u という SELinux ユーザに関連付けられていた場合，彼らがログインした際のデフォルトの SELinux ユーザは staff_u ということになる．

同時に，SELinux ユーザには Role と設定可能な Sensitivity range の範囲が設定されており，さらに「ドメイン遷移と RBAC」の節で説明した通り，Role は Domain に関連付けられている．したがって，ある UNIX ユーザに関連付けることのできるセキュリティコンテキストは，ある一定の範囲に絞られる．SELinux はある特定の UNIX ユーザが選択可能なセキュリティコンテキストの一覧を返却する API を有しており，認証デーモンはこの一覧の中から，利用者のログイン時に割り当てるセキュリティコンテキストを 1 つ選択する．

通常，利用者がシステムにログインする際，選択可能なセキュリティコンテキストからどれか 1 つを選ぶといったステップを踏む必要はない．これは，SELinux ユーザごとのデフォルト値が設定ファイルに記述してあり，認証デーモンがそれを参照するためである．ただし，認証デーモンが対応していれば，明示的に指定することもできる．

図-11 にセキュリティコンテキストを明示してログインする例を示す．Fedora や Red Hat Enterprise Linux の ssh デーモンは独自に拡張されており，利用者が複数のロールに紐付けられている場合には，ログイン時に利用するロールを選択することができる．図-11 の例では，太字で示した「/staff_r」をユー

ザ名の直後に記述することで，staff_r ロールでログインしている．このような仕組みを用いることで，たとえばサーバ管理業務を行うつもりがないのに管理者権限を持ったままログインするなど，必要以上の権限を持ってシステムを使う必要がなくなる．

まとめ

はじめに述べた通り，SELinux は他のサブシステムに対してセキュリティサーバとして振る舞う．Linux カーネル内の各サブシステムは，LSM を通して SELinux を呼び出す．これがアクセス制御の "where" に相当し，次に，呼び出された SELinux は TE・RBAC・MLS の各アクセス制御モデルを組み合わせて意思決定を行う．これがアクセス制御の "how" に相当する部分で，セキュリティコンテキストによる利用者・オブジェクトの識別と合わせて，SELinux を特徴付ける要素である．

ファイル名や利用者 ID といった，我々に馴染みのある概念を用いてアクセス制御の対象を識別するのではなく，特定のサブシステムに依存しないセキュリティコンテキストという形式で利用者・オブジェクトを識別することにより，Linux カーネルに限らず，幅広い領域に SELinux のアクセス制御モデルを適用することが可能になっている．たとえば，X Window System には SELinux が統合され，カット＆ペーストバッファを介したプロセス間通信を制御することが可能である．ほかにも，RDBMS（PostgreSQL）へも SELinux の統合が進行中であり，Apache への SELinux の統合（mod_selinux）と合わせて，最小特権下で Web アプリケーションを実行することが可能となる見込みである．

SELinux は TE・RBAC・MLS という 3 つの異なった特性を有するアクセス制御モデルを提供しているため，利用者は必要に応じて適当なセキュリティモデルを組み合わせて利用することができる．たとえば，TE はアプリケーションプログラムの特性に応じた最小特権を定義することは得意だが，利用者ごとにアクセス可能な範囲を区切ることには不向きである．こういった場合には，TE で頑張るよりもむしろ MLS/MCS を活用すべきであろう．

本稿では，SELinux がセキュリティポリシーに基づいたアクセス制御の意思決定を行い，それをオブジェクト・マネージャと連携することで，利用者の権限がどのように制約されるのかを解説した．紙面の都合により，一部機能の紹介を捨象することになったのは残念ではあるが，本稿が SELinux をより理解する助けとなれば幸いである．

（平成 22 年 7 月 27 日受付）

■海外浩平　kaigai@ak.jp.nec.com

筑波大学大学院経営・政策科学研究科卒業．修士（ビジネス）．2003 年より日本電気（株）勤務，OSS/Linux の開発・サポート業務に従事する．主に SELinux を中心としたセキュリティ機能の開発に取り組み，現在は，RDBMS に強制アクセス制御を付加する SE-PostgreSQL のメインライン化に取り組んでいる．