# SELinux を用いたビジネス・プロセスベースの強制アクセス制御による権限管理方法

岡上 智也
工学院大学大学院 工学研究科情報学専攻
E-mail: em06003@ns.kogakuin.ac.jp

小野 諭
工学院大学 情報学部
E-mail: s-ono@cc.kogakuin.ac.jp

**概要**

企業情報システムは，業務の効率化と柔軟性や会計規定へのコンプライアンスなどをめざし，IT 技術，とりわけ Web Service を活用する方向に向かっている．情報システムは，今後日本版 SOX 法に基づき IT 統制を提供する必要があり，WS ベースのシステムにおいては，分散している Web Application 毎に，業務遂行上必須な最小特権の設定とその強制が必須となる．本論文ではユーザの ID に加えてビジネス・プロセス上のコンテキストを考慮して，リクエスト毎に最小特権を付与する新たな方式を提案する．また，代表的な強制アクセス制御環境である SELinux の権限管理機能を用いて上記方式を実装する方法について述べる．本方式を用いると，従来よりも細かい粒度で，かつ業務内容に即した権限管理が可能になる．

## 1. はじめに

近年の社会の情報化により扱う情報が紙の情報から電子データの情報へと急速に移行している．今まで紙で処理をしていた信頼情報を含む情報は，効率化やデータ管理，コンプライアンスの観点により電子データのみの情報の処理に替わろうとしている．例えば企業の契約書や見積書など，これまで紙での保管が義務付けられていた法定保存書類は，e-文書法[1]により一部の例外を除き電子情報での保管が法的に認められるようになった．しかし，電子データは複製も改竄も容易なため，紙で裏付けられていない電子データのみの情報が本当に主張されている通りの正当なものかを事後的に第三者が検証するのが困難になっている．

現在情報化により企業の業務体系は Web Service (WS)[2]などの IT 技術を用いたものに移ってきている．日本版 SOX 法[3]により WS において企業の業務活動における情報の信頼性を確保する必要があり，その方法として企業が定めたポリシーを強制することによって一貫した方法で業務を行ったこと(正当性)を主張するものがある．業務活動において企業が定めたポリシーを強制させるためには，ポリシーにおいてアクセス制御記述が最小特権になっているということと，そのアクセス制御記述がアクセス制御機構によってユーザに強制することができるということが必要となる．

最小特権を実現するためには，割り当てる権限が適切な粒度で最小の権限になっているアクセス制御記述が必要となる．また通常 WS において最小特権であるアクセス制御記述を割り当てるためには，サーバなどのサービスアカウントへの権限の制限やユーザのリクエスト毎への権限の偽装(Impersonation[4]のこと),信頼コードを経由しての保護データへのアクセスなどの手順を取る．

一方，この最小特権を満たすアクセス制御記述を各 WS において強制するためには,アクセス制御記述で決められていること以外はできないことや，アクセス制御記述を強制するアクセス制御機構が十分に検証された信頼コードである必要がある．また，アクセス制御記述を強制できていたことを事後的に第三者に検証できる必要がある．アクセス制御を行う環境がこれらの性質を満たしていなければ，企業が定めたポリシーに基づいたアクセス制御記述を強制していたと主張することができない．

例えばアクセス制御記述が最小特権にする方法として，シングルサインオン (SSO) [5]により付与される ID やその役割 (ロール) と対応してセキュリティコンテキストを生成し，アプリケーションのリクエスト処理時に用いる方法[6]がある．この方法では，ユーザ ID に対応したセキュリティコンテキストはユーザが許可されている権限だけに結び付けられ，各アプリケーションにおいては，リクエスト処理時にこのセキュリティコンテキストを用いて偽装することにより最小特権を付与する．この方式を権限の粒度という観点で見ると，ID・役割粒度での最小特権が実現できていると言える．権限の粒度について考えた場合，権限の細かさと権限管理の手間とのトレードオフの問題など様々な問題が生じてくる．

本発表では企業の業務活動において適用されているビジネス・プロセス (BP) [7]に注目し，WS におけるビジネス・プロセスを考慮した最小特権アクセス制御の方式を提案する．この方法では，SSO によって付与されるグローバルな ID やロールと BP とを結び付けることにより，BP ワークフローの位置やそこ

でのオペレーション毎に新しいセキュリティコンテキストを生成する．このセキュリティコンテキストを用いることにより，BP ワークフローの位置やオペレーションによって異なる権限を割り当てることができる．これにより従来よりも細かい粒度で，より業務に即した権限管理が可能となる．また，この方式の実装環境として SELinux[8]を用いる．SELinux は，本方式を実装する上で必要となる性質である強制アクセス制御[9]や細かいアクセス制御を提供しているため,本方式の実装において必要な環境である．この SELinux において権限割り当て機能を提供しているかについて評価し，SELinux における本方式の実装方法について示す．

章２では WS のアーキテクチャと最小特権ための要件の詳細を述べ，アクセス制御記述を最小特権にするための技術として RBAC（Role Based Access Control）[6]について述べる．また，最小特権を満たすアクセス制御記述を強制する機能の観点により，任意アクセス制御[9]と強制アクセス制御について述べる．章３では権限の粒度について述べ，提案方式であるビジネス・プロセスを考慮した最小特権アクセス制御記述の方法について述べる．章４では最小特権を実現するためのプラットフォームとして SELinux について述べ,SELinux の権限管理機能について評価する．章５では SELinux を用いた最小特権付与方法の実装方法について述べる．章６では本論文についてのまとめ，今後の課題について述べる．

## 2. WS のアーキテクチャと最小特権

### 2.1 WS を使った業務活動のアーキテクチャ

企業の業務活動では WS を使って既存のサービスを組み合わせ，柔軟で業務活動にあったシステムを構築している.この WS の一般的なものとして次のようなアーキテクチャがある（図 1）.

このアーキテクチャの構成要素にはポータル,SSO ディレクトリ，ポリシーディレクトリ，実行エンジン，Web Application（WA）がある．ポータルとはユーザが WS を利用するときに最初に操作を行う場所である．SSO・ユーザディレクトリとは，WS にログインする時の認証だけで，各サービスにもログインできるようにするための SSO のためのシステムである．ポリシーディレクトリとは，WS において守るべきポリシーが書かれたポリシーファイルが入っているディレクトリである．実行エンジンはユーザの要求とポリシーディレクトリのポリシーに基づき，どのサービスに処理を依頼するのかを理解して，サービスに処理を依頼するものである．WA は WS を構成するサービスそのものであり，具体的な処理は WA を用いて処理をする．

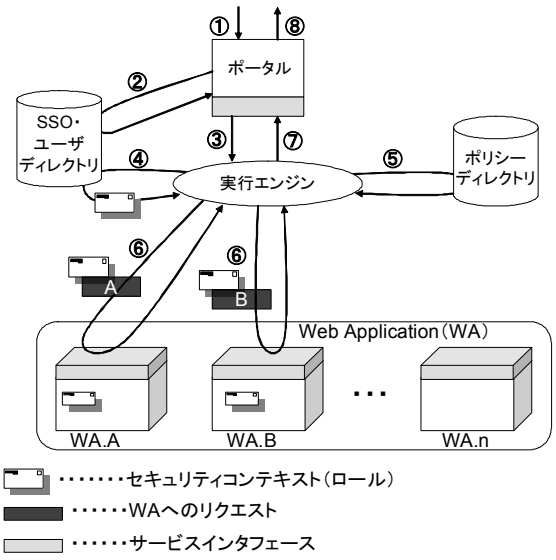


**図１　WS を使った業務活動のアーキテクチャ**

次にこのアーキテクチャにおける基本的な処理の手順について述べる．

①まず，処理要求者であるユーザはポータルより WS にログインする．

②ユーザは SSO システムにより認証を受ける．

③ユーザは実行エンジンに処理を依頼して，実行エンジンが処理を開始する．

④実行エンジンは SSO・ユーザディレクトリから，ユーザの認証情報を得る．ここでの技術として SAML（Security Assertion Markup Language）[10]があり，実行エンジンはユーザの認証情報であるセキュリティコンテキストを得る．

⑤実行エンジンはポリシーディレクトリを参照し，ユーザの要求に対してどのサービスに依頼するのかを決定する.このフロー決定のポリシーとして，業務活動ではビジネス・プロセス（BP）がある．BP は「どのようにして業務を行うかについて決めたルール」である．通常，業務を行う際にはこの BP に沿って進めていく．

⑥実行エンジンはポリシーに従って WA に処理のリクエストを送り，結果を受け取る．この時実行エンジンは，WA へのリクエストにセキュリティコンテキストを貼り付け，このセキュリティコンテキストにより各 WA でのアクセス制御を行う．

⑦実行エンジンは,必要となる全ての WA への処理を依頼し，最終的な結果を受け取るとポータルに結果を送る．

⑧ユーザに結果が返される．

## 2.2 WSにおける最小特権

WS において最小特権を実現するための要件について述べる．WSにおいて最小特権を実現するための要件として次のものを考える．

**A. アクセス制御のルールであるアクセス制御記述が，主体に対して最小の権限になっている**

**B. アクセス制御記述の実行手段が提供されている**

B-1. 全権アカウント権限からサービスアカウント権限への制限

B-2. サービスアカウント権限からリクエストアカウント権限への偽装（Impersonation）

B-3. 信頼されたコードの経由による保護データへのアクセス

**C. アクセス制御記述が強制されている**

C-1. アクセス制御記述で決められていること以外はできない

C-2. ポリシーを強制するのは，OS 組み込みのアクセス制御機構や，バグが無いことを十分に検証された信頼コードのみ

C-3. アクセス制御により強制できていたことが事後的に第三者に検証できるようなログ・監査機能を提供する

要件 A はアクセス制御記述自体についての要件である．ここで言う最小の権限とは，主体が目的としていることが行うことができ，それ以外のことが出来ないという権限である．

要件 B は，要件 A の最小であるアクセス制御記述を割り当てるための機能を提供しているということである．この機能によって，アクセス制御記述を実行する．ここでは，この機能として機能 B-1，B-2，B-3 を考える．機能 B-1 は全権を持つアカウントからサーバなどのサービスを提供するサービスアカウントの権限にまで権限を制限し，各サービスアカウントの権限で動くための機能である．機能 B-2 はサーバなどの各サービスにおいてリクエストを処理する時，リクエスト毎に権限を偽装してリクエストの権限で動くための機能である．機能 B-3 はパスワードファイルなどの保護データへのアクセスをする場合に，通常のアプリケーションには直接アクセスさせないための機能である．通常アプリケーションはバグを含む可能性があり，直接アクセスする権限があるとバグを突かれて乗っ取られた場合に非常に危険である．そのため，保護データにアクセスをする時は，バグがないことが十分に検証された信頼されたコード[11]を経由した間接的なアクセスによって実現できる．

また，アクセス制御記述を強制できなければ，いくらポリシーに沿ったアクセス制御記述でもポリシー違反ができてしまう可能性がある．そのため，アクセス制御記述を強制するために要件 C を考えた．要件 C は詳細には要件 C-1，C-2，C-3 から成る．要件 C-1 は，ポリシーが強制できていて，それ以外が禁止されているということである．要件 C-2 はバグにより，強制機構が機能しなくなることを防ぐためのものである．要件 C-3 はアクセス制御のログを残すことにより，ログから事後にアクセスが強制できていたことを証明するためのものである．これらアクセス制御記述が最小特権であることと，そのアクセス制御記述を実行する機能を提供すること，アクセス制御を強制することの全てを満たして初めて最小特権を実現できたと言うことができる．

次に図 1 の WS において，最小特権を実現できている状態について説明する．まず，主体に割り当てられる権限は，最小の権限になっている．次に WA に割り当てられる権限は，要件 B-1 により特権全てを割り当てるのではなく，必要最小限の権限のみを割り当てる．また要件 B-2 により，WA が処理をするリクエストの動作権限が，リクエスト毎にユーザが許可されている最小特権でのみ動作することができる．要件 B-3 により，パスワードファイルなどの保護されたデータには，直接アクセスできないようになっている．保護データへのアクセスは，特権を持つ信頼コードを経由してのみアクセスすることができる．そして要件 C により，これらの権限の割り当てが強制されており，ポリシーによって決められていること以外のことが禁止されている．最小特権で動作し，それが強制できていたというログを取得することができる．上記を満たした状態が，WS において最小特権が実現できている状態だと言える．

## 2.3 最小特権に関する既存技術

節２．１で示した WS において使われている，最小特権を実現するための既存技術について述べる．

WS において，要件 A である最小特権であるアクセス制御記述を記述するための技術として RBAC（Role Based Access Control）[6]がある．RBAC はユーザをロールというグループに所属させ，このロールを主体の権限としてアクセス制御を行う．従来ユーザやユーザの許可された権限が変化した場合，アクセス制御リスト（ACL）へ変更の反映に非常に手間が掛かっていた．この方法を用いるとロールに対して ACL が書かれているため，ACL におけるユーザ変更での影響が少なくて済む．これにより，ACL への反映の手間を減少させることができる．RBAC ではユーザをロールにグループ化するため，ユーザ単体での権限

の設定はできない．ただ，システムにおいて意味のあるロールを作成し権限を設定することにより，システムにおける最小特権を実現することができる．

要件 B であるアクセス制御記述を割り当てるための技術として，capability[12] (B-1) や chroot[13] (B-1)，impersonation[4] (B-2)，setuid[14] (B-3) などがある．例えば capability は，サービスアカウント権限の割り当てである．サービスを提供するサーバに全ての特権を与えて動作させるのは，悪意のあるユーザに乗っ取られた時危険である．そのため，特権をいくつもの capability という単位に分割して必要最小限の capability のみをサービスに付与することができる．

要件 C であるアクセス制御記述を強制するための技術としては，アクセス制御方式である任意アクセス制御 (DAC) [9]や強制アクセス制御 (MAC) [9]がある．DAC は従来の UNIX などの OS で使われており，MAC は主に軍用システムで扱われている．DAC はデータに対するアクセス制御の管理を，データの持ち主 (owner) が行うという特徴がある．そのため，ユーザに対して柔軟な環境を提供する一方で，システム全体で一貫したポリシーを保つことは困難である．MAC は DAC と違い，データに対するアクセス制御をデータの持ち主には任せず，システムのセキュリティを管理するセキュリティ管理者がポリシーを決めて管理する．セキュリティ管理者が定めたポリシーは一般ユーザに強制される．そのため DAC に対して環境の柔軟性は落ちるが，ポリシーに基づく一貫したシステム全体のアクセス制御管理を行うことができる．

DAC において節２．２で述べたアクセス制御記述を強制するための要件 C を満たすためには，各データの所有者がポリシーを遵守して，一貫性のあるポリシーの強制をしなければならない．DAC がデータの所有者によってポリシーを設定できるためには，一貫性を保たなければならない．MAC においては要件を満たしている．MAC のアクセス制御機構は要件 C-2 で言っているカーネルレベルであり，要件 C-3 であるアクセス制御などのログを取得することができることを満たす．また，システム管理者が決めたポリシーを全体に強制することができるという要件 C-1 を満たしている．

## 3. ビジネス・プロセスを考慮した最小特権アクセス制御記述の方法

### 3.1 最小特権の粒度

WS においてアクセス制御記述を最小にすることを考えた場合，主体のオペレーションの粒度や主体自身の判別基準の粒度が考えられる．オペレーションの粒度とは，書き込みに対する追記や上書きなどのことである．主体自身の判別基準の粒度とは，特権ユーザと一般ユーザのみ粒度から部長や課長などの細かいユーザ設定のことである．このように権限の最小性とは粒度に依存している。この粒度を細かくし過ぎると，ACL などの設定や権限管理の手間が増大してしまう．これにより権限の設定のミスの発生や，手間が膨大なために権限の粒度を無視して適当に設定してしまうなどの人的ミスを誘発させてしまう恐れがある．逆に粒度を粗くしすぎると，不必要な権限を割り当てることになり，セキュリティ上問題となる．このように権限の最小性にはトレードオフの問題があり，各環境ではうまくバランスの取れた適切な粒度である最小特権を設定する．この最小である権限の粒度について考えた場合，次の要件がある．

ⅰ．割り当てる権限に不要な権限が含まれていない
ⅱ．粒度を設定する基準の安定性
ⅲ．粒度を設定する基準の変化に対するシステムへの負荷
ⅳ．権限の粒度が意味のあるものである

要件ⅰは割り当てる権限自体が主体に対して，必要最小限の権限になっていること．要件ⅱは粒度を設定する基準が頻繁に変わらないということ．要件ⅲは粒度を設定する基準が変わった時に，システム全体への負荷や手間がどれだけ小さいかということ．最後に要件ⅳは，設定する権限の粒度が WS の環境を考慮したものであるかということである．WS を使っているユーザはあるルールに沿って処理を行っているため，WS においてこれらを定義したコンテキストが存在する．コンテキストとは，WS で決められた手順やルールの定義のことである．このコンテキストを考慮した粒度が，その WS における意味のある粒度であると言える．

ユーザ ID を粒度の基準とした場合の最小特権アクセス制御記述を考えてみる．要件ⅰに関しては満たす事ができる．これはユーザ毎に権限を設定できるため，ユーザ ID レベルで最小特権を実現することができる．要件ⅱに関しては満たしていないと言える．ユーザは比較的変わり易いため，安定していない．要件ⅲに関しては満たしていないと言える．ユーザ毎に権限を設定しているため，ユーザが変化した場合に影響する部分が多く，手間や負荷が大きい．要件ⅳに関しては満たしていないと言える．この方

法では，WS のコンテキストを考慮しておらず，WS において適切な粒度の権限とは言えない．

次に前述した最小特権アクセス制御記述の既存技術である RBAC において，粒度という観点で評価する．要件 i に関して RBAC は満たしていないと言える．RBAC では，ユーザをロールにグループ化し，ロールによってアクセス制御を行う．そのため，不必要な権限も付与される可能性がある．要件 ii に関しては満たしていると言える．ロールはユーザに比べて頻繁に変わらないため，安定している．要件iiiに関しては満たしていると言える．ユーザ ID に対してロールは少数であり，ロールが変化したときの権限管理の手間などは小さい．要件ivに関して RBAC は満たしているとは言えない．これは RBAC によるアクセス制御は，実行主体のロールだけを考慮したものであるため，WS のコンテキストを考慮していない．そのため，WS において適切な粒度である権限を割り当てることは難しい．

### 3.2 ビジネス・プロセスを考慮した最小特権アクセス制御記述の方法

企業の業務活動では，通常コンテキストとしてビジネス・プロセスのコンテキストが扱われる．ビジネス・プロセス（BP）は前述のように業務の行い方について決めたルールであり，取引の流れやメッセージの受け渡し方法，各プロセスにおける処理などが決められている．WS において BP をコンテキストとし RBAC を用いた場合，最小特権上の問題が生じる．図２を使ってこれを説明する．

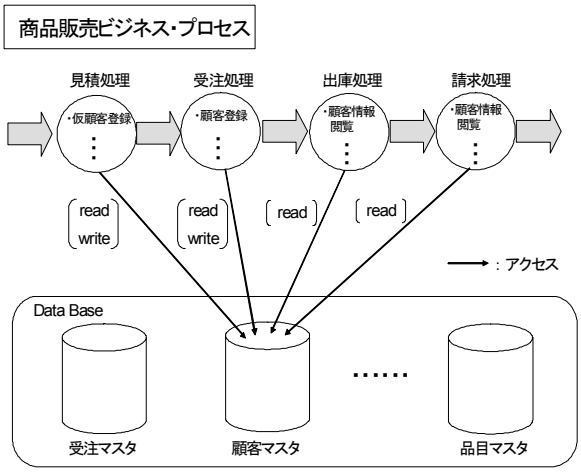


**図２　BP 基準の粒度でのアクセス権限**

図２の商品販売ビジネス・プロセスでは見積処理，受注処理，出庫処理，請求処理の４つのプロセスからなっており，左から順番に遷移していく．各プロセスでは様々な WS を扱うが，各プロセスにおいて顧客管理 WA という同一の WA を使った場合について考える．顧客管理 WA は顧客マスタというデータベースにアクセスし，顧客情報の登録や変更，閲覧などの処理を行う．BP を権限の粒度の基準とした場合，各プロセスでは顧客マスタに対して次のようなアクセス権限が許可されている．見積処理では仮顧客登録をするため[読み出し]と[書き込み]．受注処理では顧客登録をするため[読み出し]と[書き込み]．発送処理では顧客情報を閲覧するため[読み出し]．請求処理では同じく顧客閲覧をするため[読み出し]．各プロセスにおいて必要となる権限はこれだけであり，これ以上の権限の割り当ては最小特権上好ましくない．しかし，この WS においてロールの粒度でのみで権限を付与した場合，ユーザの顧客管理 WA での権限はどのプロセスにいても同じになってしまう（図３）．これはロールに対応する権限のみで，WA におけるアクセス制御を行っているためである．これでは，BP のコンテキストにおいて適切な粒度である権限とは言えない．

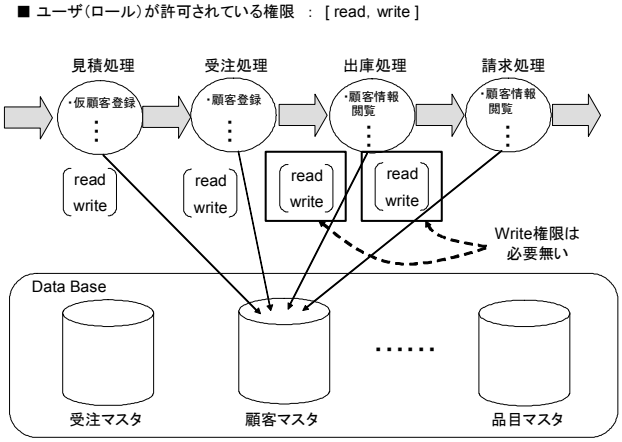


**図３　ロールの粒度におけるアクセス権限**

そこで本発表では，ビジネス・プロセスを考慮した最小特権を付与する方法を提案する．本方式は，この BP を WS のコンテキストとして用いて，SSO によるユーザのロールと BP のコンテキストの両方を考慮した権限を付与する．

本方式ではロール，プロセス，オペレーションでの粒度を考慮しており，BP の位置やオペレーションと粒度で最小特権を付与することができる．また，BP は比較的安定しており，頻繁に変更されることが無い．本方式では BP が変化した場合も，新しいセキュリティコンテキスト（BP セキュリティコンテキスト）の生成により，WA などのサービスへの影響を比較的小さくすることができる．

### 3.3 BPとロールとの結びつきによるBP粒度の権限の生成

本方式の具体的な方法として,このロールと別に,企業が定めた BP ポリシーに基づいたコンテキスト「BP コンテキスト」を用意して，ロールと BP の両方を粒度の基準とした最小特権を実現する．BP コンテキストには BP ポリシーに基づき，BP の位置やオペレーションが書かれているこれにより，現在どの BP の，どのプロセスにおいて，どういったオペレーションを行うのかを知ることができる．本方式を図４を使って説明する．まず，企業が定めた BP ポリシーに基づいた BP コンテキスト作成し,ポリシーディレクトリに格納しておく．

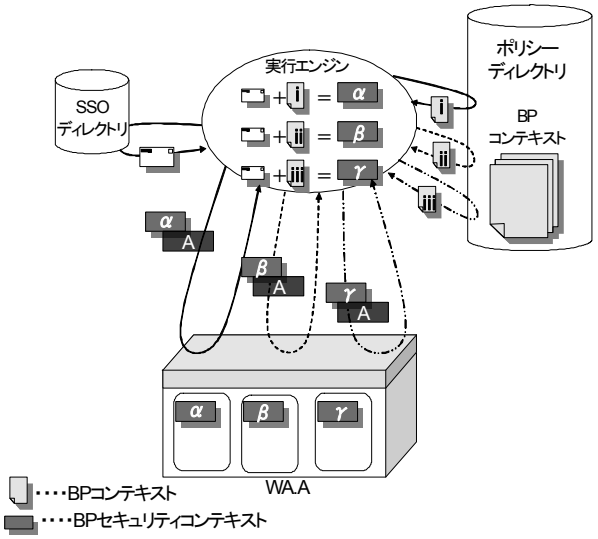


**図４　ロールと BP コンテキストの結び付けによる BP セキュリティコンテキストの生成**

ユーザは実行エンジンに処理を依頼する．実行エンジンは，SSO によりロール，またはロールに対応するセキュリティコンテキストを受け取る．次に実行エンジンはポリシーディレクトリに格納されている BP コンテキストを取得することにより,現在どの BP の，どのプロセスで，どういったオペレーションを行うのかを解釈する．今実行エンジンは，SSO によるロールとポリシーディレクトリの BP コンテキストを保持していることになる．BP を粒度の基準としているため，WA でのリクエストの権限はロール，BP，プロセス，オペレーションの全てを考慮した粒度である．そのため，実行エンジンではロールと BP コンテキストを結び付けて，あらかじめロールと BP コンテキストにより決められている新しいセキュリティコンテキスト（BP セキュリティコンテキスト）を生成する．このセキュリティコンテキストはロール，BP，プロセス，オペレーション全てを考慮した最小の権限である．このセキュリティコンテキストを WA へのリクエストに貼り付け，処理を依頼する．処理を依頼された WA では，BP セキュリティコンテキストで許可されている権限に偽装して処理を行う．これにより，BP を考慮した最小特権を実現する．

本方式とロール粒度の RBAC，ユーザ ID 粒度での最小特権の方法について比較した（表１）．

**表１　権限の粒度基準の比較**

| | 不要な権限 | 安定性 | サービスへの影響 | 意味のある粒度 |
|---|---|---|---|---|
| ユーザID | ○ | × | × | × |
| RBAC | × | ○ | ○ | × |
| 本方式 | △ | ○ | △ | ○ |

○：権限少ない，安定している，影響が少ない，意味がある
△：比較的少ない，比較的安定している，比較的影響が少ない，比較的意味がある
×：権限多い，不安定，影響が大きい，意味が無い

## 4. 最小特権を実現するプラットフォームの評価

### 4.1　最小特権を実現するための環境 SELinux

章３で述べた最小特権アクセス制御記述を実現する方法を実行できる環境は，節２．２で述べた要件 B：アクセス制御記述を割り当てる機能である B-1，B-2，B-3 の機能を提供していればよい．また，アクセス制御記述を強制できているための要件 C-1，C-2，C-3 の性質も満たしていればよい．

そこで本方式の実装環境として，SELinux（Security-Enhanced Linux）[8]に注目した．SELinux の特徴として，まず MAC をサポートしていることがある．MAC によりポリシーをシステム全体に強制することができ，ポリシーで許可されていること以外のことを禁止することができる．また，SELinux のアクセス制御機構は，LSM（Linux Security Module）というカーネルレベルでの実装となっている．これは OS の提供している信頼できるものである．また，これにより詳細なログを取得することができる．この様に SELinux の MAC では，アクセス制御記述を強制できているための要件を満たしている．

次の特徴として，粒度の細かいアクセス制御を行うことができることがある.SELinux では TE や RBAC といったアクセス制御技術を提供している．TE では対象に対する細かいパーミッションの設定や，権限の割り当てを行うことができる．これによって BP に対応した細かい最小特権を割り当てることができる．

あとはアクセス制御記述を割り当てる機能 B-1，B-2，B-3 を提供できればいいが，明確な評価例が無いため次節において評価する．

### 4.2 SELinuxにおけるDAC権限管理機能の評価

SELinuxのMAC環境は通常のLinuxへの拡張環境であるため，SELinuxでは通常のDAC環境も提供している．そのためSELinuxでは，DACとMACの両方で許可されたアクセスのみ許されている．まず，SELinuxのDAC環境における権限管理機能について，前述であるアクセス制御記述を割り当てるための機能を提供しているかについて評価する．

SELinuxにおけるDACの権限管理機能で機能B-1を提供する機能として,前述であるcapabilityがある．また，機能B-3はsetuidによって提供されている．SELinuxにおけるDACの権限管理機能で機能B-2を提供しているものは無かった．

### 4.3 SELinuxにおけるMAC権限管理の評価

次にSELinuxのMAC環境における権限管理機能について評価する．MAC環境の権限管理機能としてドメイン遷移[15]，setexeccon[16]，setcon[17]，拡張機能のsetforkcon[18]がある．

ドメイン遷移とは，Type Enforcement（TE）[15]によりプロセスに付与されるドメインという権限を，ポリシールールに従って遷移させる機能である．TEとはSELinuxで提供されているアクセス制御技術である．ドメイン遷移では，あらかじめSELinuxポリシーにおいてエントリ・ポイントとなる実行ファイルを決めておく．このエントリ・ポイントを実行することにより，実行先プロセスのドメインを設定されたドメインに遷移させることができる．

setexeccon とはexecve先のドメインを変化させる機能である．プログラムを実行する時にexecveを用いて実行し，実行先のドメインを指定したドメインに変化させることが可能である．setexeccon を使うためには，setexeccon権限とドメインが遷移できるための許可がいる．

setcon はプロセス自身のドメインを動的に変化させる機能である．setconはsetexeccon と違いexecveによるプログラムの実行を行わないため，同じプロセスの中で動的にドメインを変化することが可能である．setconを使うためには，dyntransitionの宣言と，現状では使えないようにしている制限の緩和[付録２参照]がいる．この制約が何のためにあるのかは現状ではわからない．

setforkconはLinux Conference2004でFernando Vázquez氏らによって発表されたSELinuxの拡張機能である．これはプロセスにおいてforkの実行を行うことにより，プロセス自身のドメインを変化させる機能である．execveがいらないため，軽い負荷でドメインを変化することが可能である．

それではこの４つのMAC環境の権限管理機能について，アクセス制御記述を割り当てるための機能を提供しているかについて評価する．

**■ドメイン遷移**

ドメイン遷移では，機能B-1を提供することができる．遷移先のドメインの権限をサービスアカウント権限にしておけば，エントリ・ポイントとなる実行ファイルを execve することにより全権アカウント権限からサービスアカウント権限への権限の制限が可能となる．機能B-2は提供することができない．これはドメイン遷移が execve によって遷移するためである．サービスアカウント権限からリクエスト権限への権限の偽装をするとき，リクエスト毎にexecve による起動をすると相当な負荷を与えることになり現実的ではない．機能B-3は提供することができる．ドメイン遷移はあらかじめエントリ・ポイントの指定し，静的なドメインの遷移しかできない．そのため，エントリ・ポイントに信頼コードの実行ファイルを選び，そのコードを実行した時のみ保護データにアクセスすることができるように特権ドメインに遷移するように設定すれば機能を満たすことができる．

**■setexeccon**

setexeccon では，機能B-1を提供することができる．ドメイン遷移と同様setexeccon による変更先ドメインをサービスアカウント権限に設定しておけば，全権アカウント権限からサービスアカウント権限への権限の制限が可能となる．機能B-2は提供することができない．これもドメイン遷移と同じでexecveによるリクエストアカウント権限への変更が現実的に難しいためである．機能B-3は提供することができない．setexeccon によるドメインの変更は，変更先のドメインの指定と実行する対象のセットであり，この２つは実質別々に宣言することになる．そのため，実際にはsetexeccon を使うとどの実行ファイルをexecve しても特権ドメインに変更してしまう．そのため，信頼することができないコードでも特権ドメインに変更されてしまうので，機能B-3を提供するためにsetexeccon を使うことは難しい．

**■setcon・setforkcon**

setcon では，機能B-1は提供することができる．上記２つの機能と同様に変更先のドメインを制限されたサービスアカウント権限にしておけば，サービスアカウント権限への権限の制限ができる．機能B-2は提供することができる．setconはプロセス自身のドメインを変化させるため，サービスカウントのプロセスの権限を小さい負荷でリクエスト毎のアカウント権限に偽装することができる．しかし，通

常機能 B-2 を提供できているレベルとしては，スレッドレベルでの偽装が望ましいため，setcon が完全にこの機能を満たしているわけではない．また，要件 B-3 は提供することができない．プロセス自身のセキュリティコンテキストを変化させるため信頼のあるコードでなくても権限の昇格ができてしまう．そのため，バグを使われ不正に権限の昇格を行われてしまう危険がある．setforkcon は fork と setcon を組み合わせたものと同様の機能であり，setcon と同じ言える．

それでは SELinux の全体として，アクセス制御記述を割り当てる機能を提供しているかをまとめる（表２）．SELinux では TE によるアクセス制御機能があるため，ドメインの権限を制限することにより全権アカウント権限をサーバアカウント権限に制限することができる．そのため，機能 B-1 は提供している．また，サービスアカウント権限からリクエストアカウント権限への権限の偽装は，プロセスレベルでのみ機能 B-2 を満たすため一応できる．機能 B-3 は提供している．これはドメイン遷移を使うことによって，信頼コードを経由したときのみ，保護データにアクセスできるようにすることができるためである． 以上により SELinux では，アクセス制御記述を割り当てる機能の B-1～3 をどれも満たしており（機能 B-2 は△），本発表の方式を実装する環境としての性質を十分に満たしていることがわかった．

**表２　MAC 環境における権限管理機能の評価**

| | 機能 B-1 | 機能 B-2 | 機能 B-3 |
|---|---|---|---|
| ドメイン遷移 | ○ | × | ○ |
| setexeccon | ○ | × | × |
| setcon | ○ | △ | × |
| setforkcon | ○ | △ | × |

## 5. SELinux を用いたビジネス・プロセスに基づく強制アクセス制御

### 5.1 SELinux による最小特権アクセス制御記述の実装

本発表の方式を SELinux によって実装し，WS を使った業務活動においてポリシー（BP）を強制する方法について述べていく．

本発表の提案である，BP を考慮した最小特権アクセス制御記述の方法について BP コンテキストの作成，ドメイン設計，SELinux のポリシーへの反映，実行エンジンの構築といった順番で実装していく（図５）．各ステップについて詳しく述べていく．

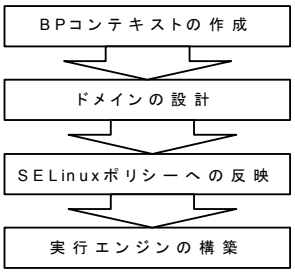


**図５　SELinux を用いた本方式の実装の手順**

**■BP コンテキストの作成**

まず，BP コンテキストを作成する．BP コンテキストは前述のように，現在どの BP の，どのプロセスにおいて，どういったオペレーションを行うのかを知ることができるコンテキストである．この BP コンテキストを，企業が定めた BP に基づいて作成する．BP コンテキストは複数種あってよい．例えば，BP のみを書いた BP コンテキストやプロセスのみの BP コンテキストなどである．

**■ドメインの設計**

次にドメインの設計を行う．ここでは主体に与える権限について設計を行う．ロールに対する権限や各 WA の権限などを考える．特に BP に関する権限の設計として，BP コンテキストに対応する最小特権となる権限を設計する．この権限は BP に基づいて設計され，かつロール毎，BP 毎，プロセス毎，オペレーション毎で最小特権となっていなければならない．このステップの注意する点として，実行エンジンの権限の問題がある．実行エンジンは SSO のロールと BP コンテキスト結び付けて，BP セキュリティコンテキストを生成する．他のアプリではこの処理ができないようしなければならない．そのため，実行エンジンに BP セキュリティコンテキストを生成できるための特権を割り当てる（図６）．この方法として，実行エンジンに節２．２の権限を割り当てるための機能 B-3：ドメイン遷移を適用しなければならない．この設計を行う．

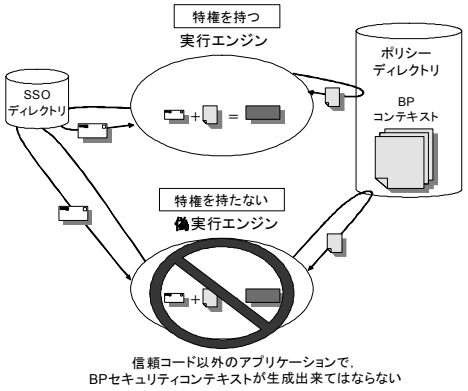


**図６　実行エンジンが特権を持つ必要性について**

■SELinux のポリシーへの反映

SELinux のポリシーの設定である．前のステップではドメインの設計をしたが，このステップでは，それを SELinux のポリシーに反映するために，ドメイン設計に沿って SELinux ポリシーを設定していく．SELinux のポリシーによりこの環境での記述したポリシーは強制されるようになる．また，前述の実行エンジンに一時的な特権を与えるために，機能 B-3：ドメイン遷移の設定をする．

■実行エンジンの構築

最後に実行エンジンの構築である．実行エンジンは BP を解釈して，どの WA を使って処理をするのかを決める．この実行エンジンに BP コンテキストの取得と，SSO によるロールと BP コンテキストを結び付けることによる BP セキュリティコンテキストの生成機能を新しく組み込む．セキュリティコンテキストの生成機能は，ロールと BP コンテキストの組み合わせに合致した権限が書かれているファイルを解釈する．このファイルの設計はドメイン設計により行われる．また，この実行エンジンは特権を持って BP セキュリティコンテキストを生成するため，できるだけバグが無いことを検証できる環境である．そのため，節２．２の要件 C-2 を満たすように構築する．

### 5.2 プロトタイプの実装

実験を行うために，本発表方式のプロトタイプを作成した．今回実装した場所は，実行エンジンが一時的に BP セキュリティコンテキストを生成することができる特権に昇格し，ロールと BP コンテキストからセキュリティコンテキストを生成する箇所である．図７に実行エンジンを使った BP セキュリティコンテキストの生成している例を示す．

```
$ id
context=guest:sale_r:sale_t
$ bpsh sale_ts1
Your old context was guest:sale_r:sale_t.
Your current context before setcon is
  guest:sale_r:bpsh_t.
target security context before setcon:
  guest:bpsh_sale_r:bpsh_sale_t
setcon ans 0
Your current context after setcon is
  guest:bpsh_sale_r:bpsh_sale_t
target security context before setexeccon:
  guest:bpsh_sale_r:bpsh_sale_ts1_t
setexeccon ans 0
```

図７　実行エンジンの実行例

ユーザ guest はシステムにログインすると，guest:sale_r:sale_t というセキュリティコンテキストが割り当てられる．これは実装環境である SELinux が割り当てたセキュリティコンテキストであるが，WS における SSO のロールと考える．この権限では，どんな業務アプリケーションにおいても業務を行うことはできない．そのためユーザ guest は，実行エンジンである「bpsh」に処理を依頼する．今回 BP コンテキストの実装まで行うことができなかったため，ユーザが「sale_ts1」という BP コンテキストの指定を行っている．ユーザが実行エンジンを実行すると，ドメイン遷移により BP セキュリティコンテキストを生成することができる「bpsh_t」という特権ドメインに遷移する．次に実行エンジンは setcon を用いて汎用のセキュリティコンテキスト「bpsh_sale_r:bpsh_sale_t」に変化する．実行エンジン bpsh は，BP コンテキストを解釈して処理を依頼するアプリケーションを決定する．そして，setexeccon を用いてアプリケーションを実行する．setexeccon によりアプリケーションでの権限は，ロールと BP コンテキストの組み合わせで最小特権となるドメイン「bpsh_sale_ts1_t」に変化させられる．これが BP セキュリティコンテキストであり，このドメインにより BP を考慮した最小特権が実現されている．

## 6. まとめ

本論文ではユーザのロールに加えてビジネス・プロセス上のコンテキストを考慮し，リクエスト毎に最小特権を付与する新たな方式を提案した．また，代表的な強制アクセス制御（MAC）環境である SELinux の権限管理機能を用いて上記方式を実装する方法について述べた．本方式を用いると，従来よりも細かい粒度で，かつ業務内容に即した権限管理が可能になることを示した．

今回 SSO によるロールと BP コンテキストを用いて BP を考慮したリクエスト毎の BP セキュリティコンテキストを付与した．これにより，BP を考慮したリクエスト毎の最小特権を実現する方法について説明した．

また，MAC 環境である SELinux での最小特権を実現するための機能について検討を行った．SELinux ではドメイン遷移や setexeccon，setcon，setforkcon などの権限管理機能があるが，保護データにアクセスするための一時的な権限昇格に使える機能はドメイン遷移だけであった．

今後の課題として，WS を構築したプロトタイプの作成がある．今回実装したのは実行エンジンの場所のみであったため，WS としての実験を行わなかった．そのため，WS の運用性を考慮した実験を行う必要が

ある．次に BP コンテキストの設計の問題がある．BP コンテキストをどのように設計するかによって，BP が変化したときに生じる BP コンテキストの変更の手間が大きいか小さいか変わってくる．BP コンテキストの変更の手間を考えた，BP コンテキスト設計が重要となってくる．また，Fedora Core 5 では，SELinux に変更が加えられているため，それに対応できるか検討する必要がある．

## 参考文献


[1] @IT 情報マネジメント用語事典　e-文書法
http://www.atmarkit.co.jp/aig/04biz/ebunshoho.html
[2] IT 用語辞典 e-Words　Web サービス
http://e-words.jp/w/WebE382B5E383BCE38393E382B9.html
[3] 企業会計審議会 著：“財務報告に関わる内部統制の評価及び監査の基準（公開草案）”，金融庁，2005.
[4] @IT　プログラミング ASP.NET
第 17 回　ASP.NET における認証と認定
http://www.atmarkit.co.jp/fdotnet/aspnet/aspnet17/aspnet17_03.html
[5] IT 用語辞典 e-Words　SSO
http://e-words.jp/w/SSO.html
[6] IPA　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
http://www.ipa.go.jp/security/fy16/reports/access_control/documents/PolicyModelSurvey.pdf
[7] @IT　ビジネスを可視化するモデル記述言語「BPMN」
http://www.atmarkit.co.jp/farc/rensai/bpmn01/bpmn01.html
[8] NSA　Security-Enhanced Linux
http://www.nsa.gov/selinux/
[9] 土居範久 他著：“情報セキュリティ事典”，共立出版，2003.
[10] @IT　強力な SSO を実現する XML 認証・認可サービス
http://www.atmarkit.co.jp/fsecurity/rensai/webserv04/webserv01.html
[11] R. Shirey 著：“RFC2828 Internet Security Glossary”，IETF，2000.
[12] @IT　セキュア OS「LIDS」入門　第 3 回
権限を最小化する Linux カーネルケーパビリティ
http://www.atmarkit.co.jp/fsecurity/rensai/lids03/lids01.html
[13] 日本の Linux 情報　JM Project chroot
http://www.linux.or.jp/JM/html/GNU_sh-utils/man1/chroot.1.html
[14] IPA セキュア・プログラミング講座
http://www.ipa.go.jp/security/awareness/vendor/programming/b07_03_main.html
[15] 中村雄一 他著：“SELinux 徹底ガイド”，日経 BP 社，2004.
[16] IPA　強制的アクセス制御に基づく Web サーバーに関する調査・設計
http://www.ipa.go.jp/security/fy16/reports/mandatory_access_contol/documents/web_server_study.pdf
[17] George Washington University　SELinux Programming
http://www.ensl.cs.gwu.edu/resources/SELinux_Programming
[18] Fernando Vázquez，保理江 高志，原田 季栄：“The need for setuid style functionality in SELinux environment”，2004.


## 付録

■SELinux のディレクトリ構成

今回実装に用いた環境として，

OS　　：Fedora Core 4 (2.6.14)
CPU　：Intel Pentium 4　3.00GHz
Memory：1GB

を用いた．この環境における SELinux のディレクトリ構成を図８に示す．

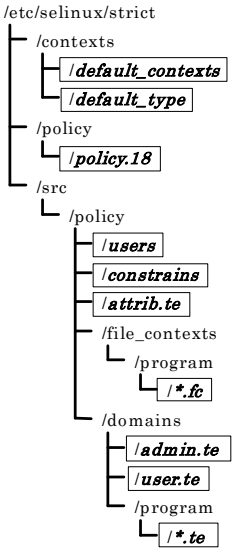


**図８　SELinux のディレクトリ構造**

■変更した SELinux ポリシーソース

権限昇格のための実行エンジンの実装にあたり，変更した SELinux ポリシーソースの一部を示す．全てのソースは/etc/selinux/strict 以下の場所である．

**－　/src/constrains**

```
constrain process dyntransition
          ( u1 == u2 and r1 == r2) or
          ( t1 == privsetcon and
            ( ( u1 == u2 or t1 == privuser ) or
              ( r1 == r2 or t1 == privrole ) or
              ( t1 == privuser and t1 == privrole ) ) );
```

**－　/src/attrib.te**

```
attribute privsetcon;
```

**－　/src/policy/domains/user.te**

```
allow bpsh_t bpsh_sale_t:process dyntransition;
allow bpsh_t bpsh_reserv_t:process dyntransition;
```