平成２０年度
「セキュア・プラットフォームに関する技術動向」
調査報告書（統合アクセス制御編）

平成 **21** 年 **3** 月

社団法人　電子情報技術産業協会
セキュア・プラットフォーム推進コンソーシアム

序

我々は平成１９年１０月にセキュア・プラットフォームの開発支援のために、セキュア・プラットフォーム推進コンソーシアムを設立した。このコンソーシアムには、普及広報 WG やユーザニーズ WG とともに、技術動向や標準化調査するための技術・標準化動向 WG を設置し、平成１９年１０月から平成２１年３月にかけて、本報告書の作成と報告会を実施した。

平成１８年度と平成１９年度に、「セキュア・プラットフォームの技術動向と標準化動向」という IT トレンドを調査したが、平成２０年度もセキュア・プラットフォームに関する新たな動向に注目し、継続調査した。本調査研究では、情報システムのプラットフォームの国際標準化とセキュリティ向上という観点から、利用者の視点からの要望などに応えるための仮想化技術、セキュリティ技術、および標準化団体やオープンソースに関する動向について調査し、現状の技術や最新動向を整理した。

約１年ではあったが、WG 委員ならびに関係者が会議を重ね、報告書をとりまとめることができた。本報告書の作成にあたり、調査研究にご協力・ご尽力頂いた関係各位に改めて深く感謝の意を表す。本報告書は、企業におけるより一層の IT 利活用の促進、さらに我が国 IT 産業の競争力強化に対して役立てられ、我が国全体の産業競争力につながることを期待したい。

平成２１年３月

セキュア・プラットフォーム推進コンソーシアム

技術・標準化動向 WG

主査 真矢　譲

－この報告書は、経済産業省から技術研究組合　超先端電子技術開発機構が受託した「平成１９年度セキュア・プラットフォームプロジェクト」の一環として作成した。－

技術・標準化動向 WG 構成メンバ

(敬称略・順不同)

主　査	真矢　譲	株式会社日立製作所
副主査	小口　芳彦	富士通株式会社
委　員	新村　昭好	沖電気工業株式会社
委　員	楯岡　正道	東芝ソリューション株式会社
委　員	川戸　正裕	日本電気株式会社
委　員	中江　政行	日本電気株式会社
委　員	前野　義晴	日本電気株式会社
委　員	稲越　宏弥	富士通株式会社
委　員	江崎　裕	富士通株式会社
委　員	恩塚　新治	富士通株式会社
委　員	酒井　敦	富士通株式会社
委　員	長谷部　高行	富士通株式会社
委　員	片山　吉章	三菱電機株式会社
委　員	國分　俊介	三菱電機株式会社
事務局	平岩　信明	社団法人　電子情報技術産業協会

－目　次－

1. はじめに

平成１８年度、社団法人　電子情報技術産業協会は経済産業省からの委託により「セキュア・プラットフォームに関する調査」を実施した。この調査では、現在のシステムプラットフォームに求められる重要な課題として、「サーバ統合による運用効率化」と「内部統制・セキュリティの強化」に整理した。また、これらの課題を解決するため、必要な技術開発の動向、および技術の普及発展に重要な役割をはたすオープンソース化と標準化活動の動向を調査し、全体像を把握した。

本年度の調査活動では、昨年度の調査をベースに、システムプラットフォームにおける課題を解決するために必要な仮想化やアクセス制御などの技術動向と製品化動向、および技術の普及発展に重要な役割をはたすオープンソース化と標準化活動の動向を調査する。

本報告書では、「内部統制・セキュリティの強化」の観点から、これらの課題を解決するために必要な統合アクセス制御の技術動向について述べる。

(1) 調査項目

統合アクセス制御のためのソフトウェアの技術項目について、本調査活動では、以下の 5 つの項目に分けて整理することとした。これらを図 1.1に示す。

(a) ID 管理の技術動向

セキュリティポリシが規定するアクセス権は、サブジェクト(S)、オブジェクト(O)、アクション(A)の組(S,O,A)に対する可否であると定義できる。サブジェクトの管理を行う機能が、ID 管理である。

(b) リソース管理の技術動向

セキュリティポリシが規定するアクセス権は、サブジェクト(S)、オブジェクト(O)、アクション(A)の組(S,O,A)に対する可否であると定義できる。オブジェクトの管理を行う機能が、リソース管理である。

(c) ポリシ適用の技術動向

セキュリティポリシが規定するアクセス権は、サブジェクト(S)、オブジェクト(O)、アクション(A)の組(S,O,A)に対する可否であると定義できる。アクションも含めた組(S,O,A)に対する可否を管理・実施する機能が、アクセス制御ポリシ適用である。

(d) ログ管理（監査）の技術動向

アクセス制御が正しく行われたことを容易に確かめるための機能が、ログ管理（監査）である。

(e) 統合管理の技術動向

個々の ID 管理、リソース管理、ポリシ適用、ログ管理を統合して、内部統制・セキュリティの強化を図る機能が統合管理である。

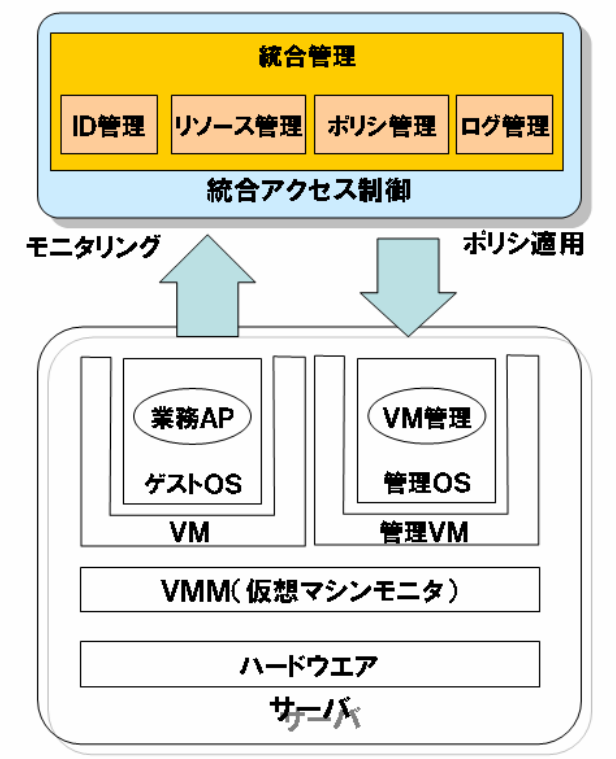

図 1.1 統合アクセス制御のためのソフトウェアの技術項目

⑵ 調査方法

上記の項目ごとに文献調査を行なうこととし、公開されている論文・技術文献・製品情報・企業/標準化団体のURL等を調査した。H20 年度の調査の狙いを表 1.1にまとめる。次章以降にその結果をまとめている。

表 1.1　H20 年度の調査の狙い

技術項目	調査の狙い	備考
ポリシ適用	RBAC モデルやポリシ記述言語についての関連製品の動向を調べ、標準仕様の採用や拡張機能について比較を行う。 RBAC ポリシとリソース使用量制限ポリシについて、記述・適用の実装技術・対応商用製品（Linux、Windows 等）の動向を調べる。	H20 年度に重点的に調査する項目
リソース管理	WBEM のプロトコル仕様の開発状況と対応商用製品（WinRM 等）、OSS（Openwsman 等）の最新動向を調べる。 CMDBf などの構成管理向けのデータベースの対応商用製品の最新動向を調べる。	H20 年度に重点的に調査する項目
統合管理	ポリシ、リソースといった個別項目だけでなく、統合アクセス制御全体の商用製品動向を調べる。	H20 年度に新規に調査する項目
ID 管理	ID ライフサイクル管理を中心に、ID 管理の仕様開発と実装技術・対応製品の最新動向を調べる。	技術的進展があったかウォッチする項目
ログ管理	大量ログの収集・分析技術について、保存期間、圧縮、検索、フォレンジックなどの観点から製品動向を調べる。	技術的進展があったかウォッチする項目

(3) 技術の位置づけ

本報告書で挙げられた技術の位置づけを、以下の表に示す。

表 1.2　統合アクセス制御技術マップ

技術項目		レベル１（~2008 年）	レベル２（2009 年に開発）	レベル３（研究）
ID 管理	ID 管理モデル	ID プロビジョニング	ID ライフサイクル管理	－
	配信プロトコル	SAML	ID-FF（ID フェデレーション）	－
	管理対象情報	ロール管理、グループ管理	－	－
リソース管理	情報表現	CIM 仮想マシン対応	CIM 共通モデル	アクセス制御モデルの標準化
	情報収集	WBEM マッピング、WBEM ディスカバリ	WBEM プロトコル（WS-Man）	－
	情報検索	WBEM クエリ言語（CQL/WQL）	CMDBf	－
ポリシ適用	ポリシモデル	階層化 RBAC	制約付き RBAC	ポリシモデル・配布機能の標準化
	ポリシ言語	XACML RBAC プロファイル	分散アクセス制御連携	－
	ポリシ適用機構	OS、VM のポリシ適用の統合	AP（MW）とのポリシ適用の統合	NW とのポリシ適用の統合
ログ管理（監査）	監査・ログ管理	監査ログ形式（XML）	監査ログ-ポリシ連携、ログ分析	大量ログ分析
統合管理			ID 管理とポリシ適用の連携	ID 管理、リソース管理、ポリシ適用、ログ管理の統合
効果（何ができるか）		ID 管理と連携したポリシ適用	管理対象の構造と時間軸にまたがる統合管理	マルチベンダでの内部統制・セキュリティ強化

2. 統合管理に関する技術動向

2.1 概観

本章では、1章で概観した4つの技術項目のうち、サーバシステム統合管理に関わる要素技術について述べる。ユーザニーズ動向調査によれば、システム統合管理の目的として「運用コスト削減」「内部統制・セキュリティ強化」等の項目があがっている。効率化・コスト削減はシステム統合に対する常に変わらない要件といえるが、近年はこれに内部統制、すなわち所定の統制ルール（あるいはセキュリティポリシ）に基づいてアクセス制御を正しく行うこと、またそれを容易に確かめうること、という要件が加わった。

一般に、セキュリティポリシが規定するアクセス権は、サブジェクト(S)、オブジェクト(O)、アクション(A)の組(S,O,A)に対する可否であると定義できる。サブジェクトとは、アクセスを行う主体を指し、ユーザIDで規定される。オブジェクトとは、アクセスが行われる対象を指し、ファイルや通信先などのリソースに対応する。アクションとは、アクセスの種類を表す。生成（create）、読込み（read）、書込み（write）、消去（destroy）などがある。サブジェクトの管理を行う機能が、ID管理である。オブジェクトの管理を行う機能が、リソース管理である。アクションも含めた組(S,O,A)に対する可否を管理・実施する機能が、アクセス制御ポリシ適用である。上記の機能は、アクセス制御を正しく行うための機能である。さらに、アクセス制御が正しく行われたことを容易に確かめるための機能が、ログ管理（監査）である。以下では、これらの要請に対応する基盤技術として、ID管理・リソース管理・アクセス制御ポリシ適用・ログ管理（監査）に注目し、具体的に説明する。

2.2 ID管理

セキュリティポリシに現れるサブジェクトとは、アクセスを行う主体となる人やプログラムを指し、ユーザアイデンティティ（user ID）で規定される。ユーザアイデンティティの管理を行う機能が、ID 管理である。通常、ID 管理は、企業におけるアプリケーションユーザを特定する識別情報と、ユーザに与える権限の管理を指す。例えば、ユーザのアカウントの作成、停止や、ユーザに与える権限の変更などの作業が含まれる。

情報システムがID管理を導入すると、さまざまな利点が得られる。

① ID管理は、コストを削減する。アプリケーションユーザの管理は、エラーの発生しやすいプロセスである。ID管理により、管理タスクを集中化・自動化することで、精度を高め、人手による労力を削減することができる。

② ID管理は、迅速なサービスの立ち上げを可能にする。新しいサービスの立ち上げには、ユーザごとにアカウントと権限を作成することが必要になる。ID管理を用いると、新しいサービスの立ち上げにも、既存のインフラストラクチャを利用してユーザ管理ができるため、立ち上げに要する時間を短縮することができる。

③ ID 管理によって、エンドユーザの利便性が改善される。ID 管理によって、ユーザはアプリケーションにすばやくアクセスでき、無駄な時間を費やすことがなくなる。すべてのアプリケーションにシングルサインオンを提供すると、ユーザが多くのログイン識別子・パスワード情報を保持する必要がなくなる。

④ ID 管理によって、アプリケーションのセキュリティが強化される。ID 管理により、ユーザのクレデンシャル情報を集中管理できる。これにより、ユーザがパスワードを書き留めておく、といった危険な行為が削減される。

アクセス制御の前提となるのは、情報システムが、ユーザを特定する識別情報とユーザに与える権限とを含む ID 情報を特定できることである。ID 管理の対象となる ID 情報は、以下の項目で構成される。

- ID 情報
  - 識別情報
    - ✧ すべてのユーザを一意に特定する、アカウント、ユーザ ID などの識別子（Unique Identifier)
    - ✧ パスワード、X.509 証明書、バイオメトリックス認証に使用される情報などのクレデンシャル
  - 権限
    - ✧ ロールやグループ（所属する組織）や識別情報の状態（有効・無効）などアクセス制御の決定に関わる情報
    - ✧ ユーザが人間の場合の氏名や住所、ユーザがプログラムの IP アドレスなどの属性の集合であるプロファイル情報

本項では、上述の概念に基づく ID 管理に関する考え方や技術動向を概説する。

2.2.1 ID 管理モデル

ID管理とは、さまざまなID情報を持つシステムへのアクセスの主体者に関するID情報を一元的に管理するものである。この一般的なモデルを図 2.1に示す。従来の情報システムでは、Webサービス、データベースサーバ、メインフレームなどで多様なID管理の方式が混在し、ID情報もOSやアプリケーションに分散して管理される形態が普通であった。これらのID情報の一括管理の実現にあたって、大きくふたつのシステムが必要となる。ひとつは、ID情報を管理するID管理サーバである。ID管理サーバでは、データベースにID情報を格納する（図中のIDリポジトリを指す)。もうひとつは、ID管理サーバで管理されたID情報とその他の情報をもとに、ユーザの識別、認証やアクセス制御を行う管理ターゲットである。管理ターゲットには、OS、データベース、LDAPディレクトリ等が想定される。また、管理ターゲットに、ID管理サーバに対応するID管理エージ

ェントが存在する場合には、SPML (Service Provisioning Markup Language)などで標準規格化された通信プロトコルを経由した管理が可能である。そうでない場合には、個々の管理ターゲットが対応できる通信プロトコルを、ID管理サーバが提供する必要がある。

ユーザという観点で見ると、ID 管理により、管理する人達（ID 管理サーバの管理者や ID 管理サーバに登録する ID 情報の変更の承認者）と管理される人達（個々の管理ターゲットの管理者や管理ターゲットに登録されたエンドユーザ）とに分けられることになる。

ID 管理サーバを構築し、ID 情報を登録する際には、会社で管理する人事情報など信頼できる情報（図中の人事マスタを指す）を元にロールやグループを一括して設定する方法、すでに稼動している個々の管理ターゲットが保持するすべての ID 情報を取り出してまとめて ID 管理サーバに格納する方法、ID 管理者が人手に個別に登録する方法といったさまざまな方法が考えられる。

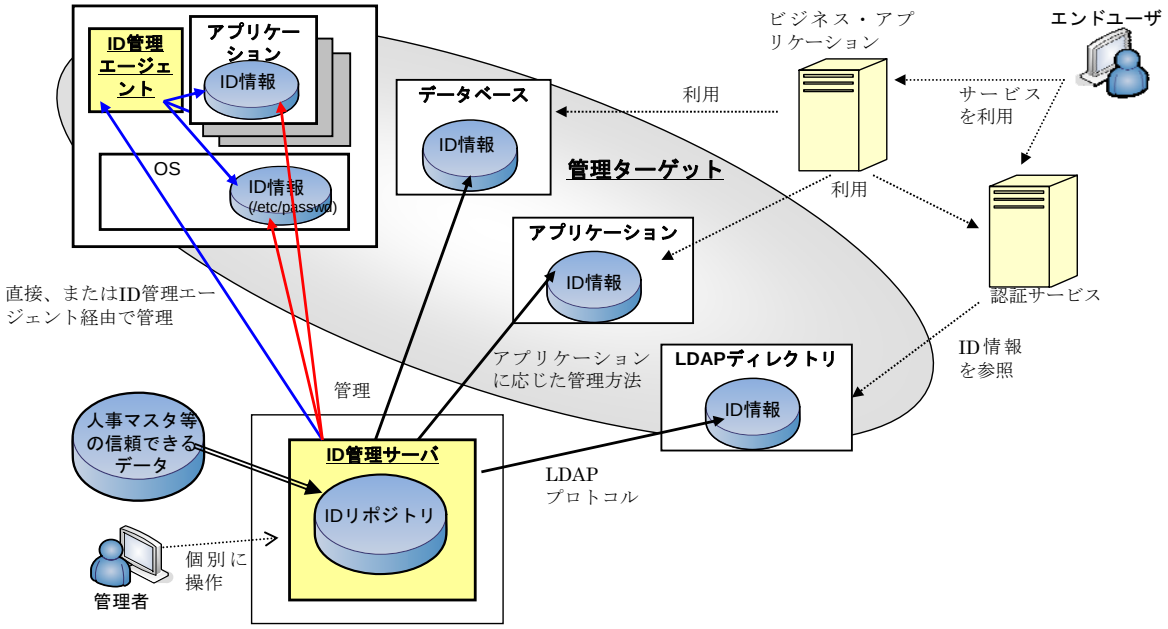

図 2.1　ID 管理の一般的モデル

2.2.2　ID 管理機能

ID 情報の管理には、ID 情報の格納、同期、ライフサイクル管理、連携の機能がある。

(1) ID 情報の格納機能

すべての ID 情報を格納し、ID 情報を一元的に管理する機能を指す。すべての ID 情報を格納する巨大な単一のリポジトリを構築する必要はなく、メタディレクトリを構築するのが一般的である。メタディレクトリとは、いくつにも分散している管理ターゲットを連携させた、仮想的なひとつのディレクトリである。メタディレクトリの肥大化を防ぐため、管理ターゲットどうしの ID

情報の紐付け情報（例：A システムのユーザ ID “aaa”と B システムのユーザ ID “taro”は同一ユーザ）を一元管理するバーチャルアイデンティティと呼ばれる技術を実現するミドルウェアも提供されている。

⑵ ID 情報の同期機能（ID プロビジョニング）

ID 情報を利用するシステムに対して ID 情報を同期し、整合性を維持する機能である。特に、マスタとなる ID 管理サーバ内のリポジトリと被管理ターゲットの ID 管理簿間の ID 情報の同期を自動的に実現する機能は、ID プロビジョニングとも呼ばれる。現在、ID プロビジョニングのためのプロトコルとしては、Open SSH や Telnet 等の汎用的な手段が用いられることが多い。ID プロビジョニングのための標準プロトコルとして普及が期待されているのは、後述する Service Provisioning Markup Language (SPML)である。

⑶ ID 情報のライフサイクル管理機能

ユーザの転勤や退職などで使われなくなった情報システムアカウントを、パスワード変更もせず放置しておくと、不正侵入者に悪用されるのは容易に想像できる。また組織内の異動や昇格にともなうアクセス権限変更を適切に行うには、ID 情報に異動や昇格などを迅速に反映する必要がある。ID 管理の実現には、これら ID 情報の申請・登録・変更・失効といった「ライフサイクル管理」を実現する機能が必要である（前述の「ID プロビジョニング」が、ID 情報のライフサイクル管理自動化を含む意味に用いられることもある）。ID ライフサイクル管理機能には、ワークフロー機能を利用し、ID 情報の作成、アクセス権限等の属性情報の変更、ID 削除等を、適切な第三者の承認のもとに行うような統制管理の実現が求められる。

⑷ ID 情報の連携機能

大規模な組織では、既に部分的に ID 管理システムの運用を開始している場合がある。ID 情報の連携機能は、複数の ID 管理システムが連携（フェデレーション）して、全体として ID 管理を実現するものである(Federated Identity Management)。これにより、既存の ID 管理システムを無駄にすることなく、負荷を分散した ID 管理が可能となる。

なお、ID 情報の連携管理では、ひとつの ID 管理システムで識別・認証後に、他の ID 管理システムと連携することになるため、ID 管理のみで完結するものではないが、連携先で識別・認証された ID に対しての ID 付与、匿名 ID 対応等の機能が必要となる。

2.2.3 Services Provisioning Markup Laguage

Service Provisioning Markup Language (SPML)は、2001 年末 OASIS に発足した Provisioning Services Technical Committee (PSTC) により策定された、ID プロビジョニングのためのオープンな技術標準（プロトコルと XML メッセージ仕様）である。2003 年に SPML1.0、2006 年に SPML Version2 (SPMLv2)が策定されている。ID プロビジョニングの操作には、ID 情報の追加、変更、

削除、検索、取得が含まれる。これらの操作を表現する XML による言語表記と操作を伝達するための通信プロトコルなどが規定されている。

SPML v2 によって ID プロビジョニングを行う主体は、次の 4 種類である。これらの主体は、SPML 標準規格の中で要素（エレメント）として記述される。

- Requesting Authority（RA)
  SPML のリクエストを発行する要素。ID 管理サーバのようなアイデンティティ・プロビジョニング・ミドルウェアを起動するアプリケーション、ID 管理管理者コマンドなどがこれに当たる。
- Provisioning Service Point (PSP)
  RA からのリクエストを受け取り、レスポンスを返す要素。例えば、ID 管理サーバのようなアイデンティティ・プロビジョニング・ミドルウェアが、この PSP に当たる。
- Provisioning Service Target (PST)
  プロビジョニングが実際に行われる要素。OS、ディレクトリ、RDB などがこれに当たる。
- Provisioning Service Object (PSO)
  PST 上のデータエンティティ。個々の ID、属性、ロール等がこれに当たる。

これら 4 者の関係を示すEntity Relationship Diagram (ERD)を図 2.2に示す[3]。

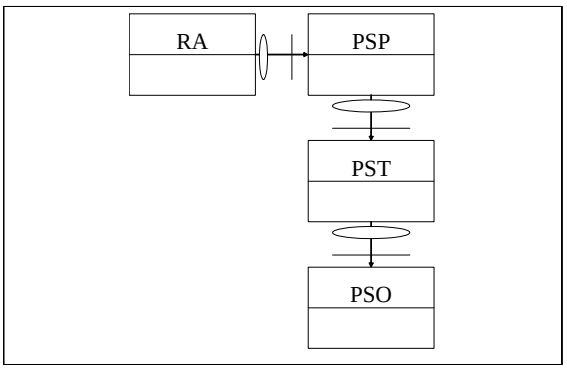

図 2.2 SPML Version2 を使う主体どうしの ERD

IDプロビジョニングの操作は、Add（追加)、Modify（変更)、Delete（削除)、Lookup（PSOの検索)、ListTargets（PSP管理下のPSTのリスト取得）の 5 種類が基本操作（Core Operations）であり、SPMLではこれらの操作を主体間で行うためのXMLメッセージの書式が規定されている。このうちSPMLv2 のAddRequestの書式の例を図 2.3に示す[3]。

```
<addRequest requestID=”127” targetID=“target2”>
<containerID ID=”ou=Development, org=Example”/>
<data>
<Person cn=”joebob” firstName=”joebob” lastName=”Briggs” fullName=”JoeBob Briggs”>
<email>joebob@example.com</email>
</Person>
</data>
</addRequest>
```

図 2.3　SPMLv2 リクエスト例（AddRequest）

これは、target2 というPSTのou=Development,org=Exampleという組織に、JoeBob Briggsという人物（Person）を追加することを要求している。これを受け取った主体は、例えば図 2.4のようなAddResponseを返す[3]。

```
<addResponse requestID=”127” status=“success">
<pso>
<psoID ID=”2244” targetID=“target2”/>
<data>
<Person cn=”joebob” firstName=”joebob” lastName=”Briggs” fullName=”JoeBob Briggs”>
<email>joebob@example.com</email>
</Person>
</data>
</pso>
</addResponse>
```

図 2.4　SPMLv2 レスポンス例（AddResponse）

この addResponse メッセージは、属性 status により、要求された追加が成功したことを要求元に通知している。

SPMLv2 の PST は、これらの基本操作のほか、Capabilities という拡張操作や拡張構文の集まりをサポート可能である。また下記に示す標準的な Capability が定義されており、これらのサポー

ト有無は PST の実装依存となっている。

- Async （操作の非同期処理モード提供、ステータス問合せ・操作キャンセル）
- Batch （SPMLv2 操作のバッチ処理）
- Bulk （条件式やワイルドカードを使った大量のオブジェクトの一括修正・削除）
- Password （特定の PSO に対するパスワード設定・失効・リセット・検証）
- Reference （PSO 同士の論理関係の問合せ）
- Suspend （処理の中断・再開）
- Search （検索および検索結果セットへの繰り返し検索）
- Updates （PSO の更新記録の検索）

SPMLv2 では、コアとなるプロトコル定義は特定のデータモデルに依存しない仕様となっているが、オプションとして、XML Schema と DSML の 2 種類のデータモデルに対応した仕様（XSD Profile、DSML Profile）も定義している。

2.2.4 SPML に関連する標準規格

ID プロビジョニングに用いられる SPML に関連する標準規格には、以下のようなものがある。

(1) Directory Service Markup Language

Directory Services Markup Language (DSML)は、OASIS の DSML Technical Committee が策定した XML ベースの LDAP ディレクトリサービス用言語であり、管理ターゲットが LDAP のときに SPML と連携して使用することが想定される。主にディレクトリ（LDAP）と Web サービスとを仲介するために利用されるものである。

(2) Security Assertion Markup Language

Security Assertion Markup Language (SAML)は、OASIS の Security Services Technical Committee が策定した XML ベースのセキュリティ用言語であり、2005 年 3 月に策定された 2.0 が最新である。連携 ID 管理でのセキュリティ通信等に使用されている。ID 管理用のプロトコルというわけではないが、セキュリティ機能のために使用されることがある。

(3) Liberty Identity Federation Framework

ID-FF:Liberty Identity Federation Framework（ID-FF)は、Liberty Alliance Project が 2003 年に策定した SAML をベースとした連携の枠組みで、これを元にした相互接続試験が行われている。（匿名での ID 連携等）ID 連携として共通に提供されるべき様々な機能の確認が行われる。

2.2.5　製品情報

既に製品化されている ID 管理製品の概要を以下に示す。

- Sun Java System Identity Manager － 関連する Sun Microsystems の製品と共に、主に大規模ユーザにおいて ID 管理に使用され始めている。
- Oracle Identity Manager － 関連する Oracle 製品と共に提供している。
- RSA Federated Identity Manager － 連携 ID 管理製品として提供している。
- Microsoft Identity Integration Server － 関連する Microsoft 製品と共に ID 管理機能を提供している。

2.2.6　まとめ

ID 管理に関連する技術動向の要点を以下にまとめる。

- ID 管理は、ユーザを特定する識別情報とユーザに与える権限とを含む ID 情報の格納、同期、ライフサイクル管理、連携の機能を提供し、ID 情報を一元管理するものである。これにより、コスト削減、迅速なサービス立ち上げ、エンドユーザの利便性改善、アプリケーションのセキュリティ強化などの効果が期待できる。
- ID 管理に関する標準規格として、ID プロビジョニングに用いる SPML が開発されている。さらに、DSML、SAML、ID-FF といった関連する規格の標準化が進められている。多数のベンダが提供する多様な管理ターゲットを一元的に管理するための相互運用性の実現にむけて、開発が進められている。

2.3　リソース管理

セキュリティポリシに現れるオブジェクトは、アクセス対象となるファイルや通信先を指すものであり、リソースと総称される。これらを管理するのがリソース管理である。典型的なリソース管理では、ハードウェア・ソフトウェアなどのリソースの識別子、属性、相互の関係などの情報を統一的なデータベースに保持し、ハードウェアの新規インストールやソフトウェアの設定変更等において、更新前と更新後の情報システムの状態を容易に把握できるようにする。広義には、リソースに関する情報を収集したり、検索したりする機能も含まれる。

情報システムがリソース管理を導入すると、さまざまな利点が得られる。

① リソース管理は、コストを削減する。大規模な情報システムには、多様なハードウェア・ソフトウェアがインストールされている。これらを棚卸しする作業を集中化・自動化することで、精度を高め、人手による労力を削減することができる。

② リソース管理は、迅速な情報システムの構成変更を可能にする。情報システムの構成変更が正確になされたことを人手で確認するには、多くの時間と労力を要する。リソース管理を用いると、構成変更の前後の違いが明確となり、確認やテストに要する時間を短縮することが

できる。

③ リソース管理によって、相互運用性が高まる。リソース管理によって、ベンダに依らない一貫的な管理が可能となる。複数のベンダからの製品を含む情報システムの運用管理者にとっての利便性が高まる。

④ リソース管理によって、リソースを一元的に集中管理することで、設定ミスやセキュリティポリシの適用し忘れを回避することができる。

情報システムが利用するハードウェア製品、ソフトウェア製品、それらを管理するリソース管理の機能を含む運用管理製品は、多様なベンダから提供されている。したがって、マルチベンダ環境での相互運用を想定した、オープン標準に基づく、リソースに関する情報表現が重要になる。本節は、最も普及している、ベンダニュートラルなリソースの概念モデルである CIM (Common Information Model)を中心に述べる。CIM は、DMTF (Distributed Management Task Force)で規定され、現在も拡張が行われている標準規格である。CIM をベースとして、情報システムを運用管理するための標準規格の集合が、Web-Based Enterprise Management (WBEM)である。CIM と WBEM により、ベンダに依存せず、情報システムに分散したリソース情報を収集し、管理者や管理システムが収集したすべての情報を検索することができる。また、仮想マシン技術に関わるリソース管理に注目し、オープンソースの VMM（Xen）で CIM を実装する取り組みである Xen-CIM についても紹介する。

2.3.1 Common Information Model

複数の計算機上で複数のアプリケーションが動作している状況を適切に把握することは、情報システムを管理する上で重要なことである。現在、広く使われているプロトコルに Simple Netowork Management Protocol (SNMP)がある。管理サーバから SNMP を用いて各計算機に問合せを行えば、各計算機の上にインストールされた SNMP クライアントからレスポンスが返り、計算機の状況を知ることができる。SNMP では Management Information Base (MIB)を用いて計算機の状況を表現しており、主にネットワークに関係する状態を管理している。

似た機能を実現する標準規格として Desktop Management Interface (DMI)がある。DMI では Management Information Format (MIF)データベースを用いて、ネットワークに限らず、管理対象の計算機のハードウェア/ソフトウェアを管理する仕組みを提供している。さらに DMI2.0 では SNMP のエージェントとして動作する仕組も設けられている。

これらをさらに進めたものが Common Information Model (CIM)である。

(1) CIM

CIMは、情報システムの構成要素（デバイス、ネットワーク、記憶領域など）をオブジェクト指向の考え方でモデル化し、SNMPやDMIの標準規格における管理の差異をなくすことを目的としている[4], [5]。マルチベンダシステムにおけるリソース管理の標準化団体DMTFにより策定さ

れ、特定のベンダに依存せずに、ネットワーク、ハードウェアまたはソフトウェアなどの計算機資源を表現するための技術標準となっている。大きくは「スキーマ（Schema）」と「仕様（Specification）」からなる。

なお、CIM は SNMP の MIB や、DMI の MIF に相当するものであり、CIM にアクセスするためのプロトコルは別途定義されることになる。

(a)　　スキーマ

CIMのスキーマ（Schema）は、モデルの定義の集合であり、コアモデル、共通モデルの 2 階層よりなる。さらに、CIMを実装する各社が独自に拡張した拡張モデルを定義できる構成となっている（図 2.5）。

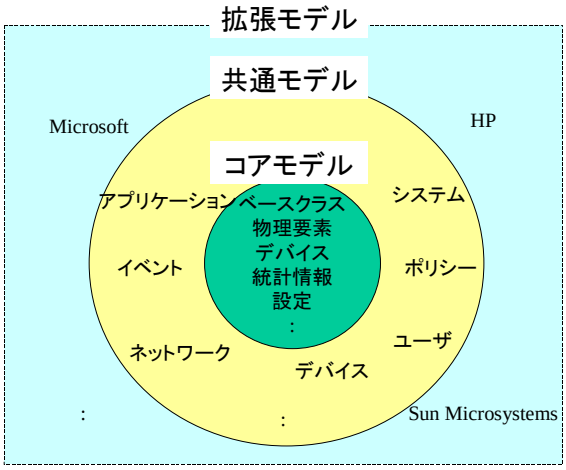

図 2.5 CIM スキーマ

コアモデルは、管理対象となるリソースが共通に持つ性質、要素などを定義するものである。ベースクラス、物理要素（ハードウェアを指す）、デバイス（共通モデルにおいてネットワーク機能を拡張定義するための基底クラスを指す）、統計情報、設定などのクラスを含む。

共通モデルは、プログラムの設計や実装に十分な詳細度のリソース管理領域を定義するものである。アプリケーション、イベント、ネットワーク、デバイス（コアモデルにおける基底クラスを拡張したクラスを指す）、システム、ポリシ、ユーザなどのクラスを含む。

Linux システム用としては Standards Based Linux Instrumentation (SBLIM)プロジェクトによる実装がある。

(b)　　仕様

CIM の仕様(Specification)は、UML をベースにしたオブジェクト指向のメタモデルで、CIM スキーマのシンタックス、ルールを規定するものである。CIM スキーマのシンタックスは、Interface Description Language (IDL)をベースにした Managed Object Format (MOF)という言語を用いて、クラス、メソッド、参照（Reference）、関係（association）などについて記述されている。

(2) Xen-CIM

CIM は一般的な計算機資源を対象にしている。しかし、仮想化された環境では、仮想化されたデバイスなど物理的に存在しない計算機資源が存在している。そこで、Xen-CIM プロジェクトでは、CIM に基づき仮想マシンモニタ Xen のホスト（Dom0）、ゲスト (DomU)のモデルを定義し実装している。Xen-CIM プロジェクトは、DMTF の SVPC (System Virtualization, Partitioning, and Clustering)ワーキンググループで定義された仮想化に関する標準規格を実装するオープンソース開発コミュニティである。外付けの機能として提供されている、検索などの CIM に対する要求を処理する Xen-CIM プロバイダの機能が、Xen 本体の実装（Xend API、 libvirt、xm）に反映される予定である。Xen-CIM を用いることにより、ホストの計算機資源を把握し、ゲストに対して適切に計算機資源を割り当てることが可能になる。

Xen-CIMにおけるCIMによるXenのモデルは、以下のようなオブジェクトより構成される[6]。

- ComputerSystem
- LogicalDevice
- SystemDevice
- ResourceAllocationSettingData

これらの関係を図 2.6に示す[12]。３つのブロックのうち、左側のブロックが、ホスト（Dom0）を表す。右側のブロックが、ゲスト（DomU）を表す。Xen-CIMにおけるゲスト（DomU）の生成とライフサイクル管理は、右側のブロックのVirtualizationManagementServiceが行う。ゲスト（DomU）のライフサイクル管理は、CIM の ComputerSystem オブジェクトが持つ RequestStateChange()メソッドにより行う。ComputerSystemは、プロセッサ、メモリ、ディスク、ネットワークなどを含むコンピュータシステムを表す。このクラスを拡張することにより、ホスト(Dom0)、ゲスト（DomU）などの多様な物理マシン、仮想マシンを表現することができる。LogicalDeviceは、ホスト(Dom0)の持つデバイス（CPU、RAM、ディスク、ネットワークインタフェースカードなど)、ゲスト（DomU）の持つデバイスを表す。SystemDeviceは、LogicalDeviceとComputerSystemの間の関係性（association）を表す。ResourceAllocationSettingDataは、Xen固有の仮想化されたLogicalDeviceの設定を表す。この設定により、リソースの割り当てを表現することができる。

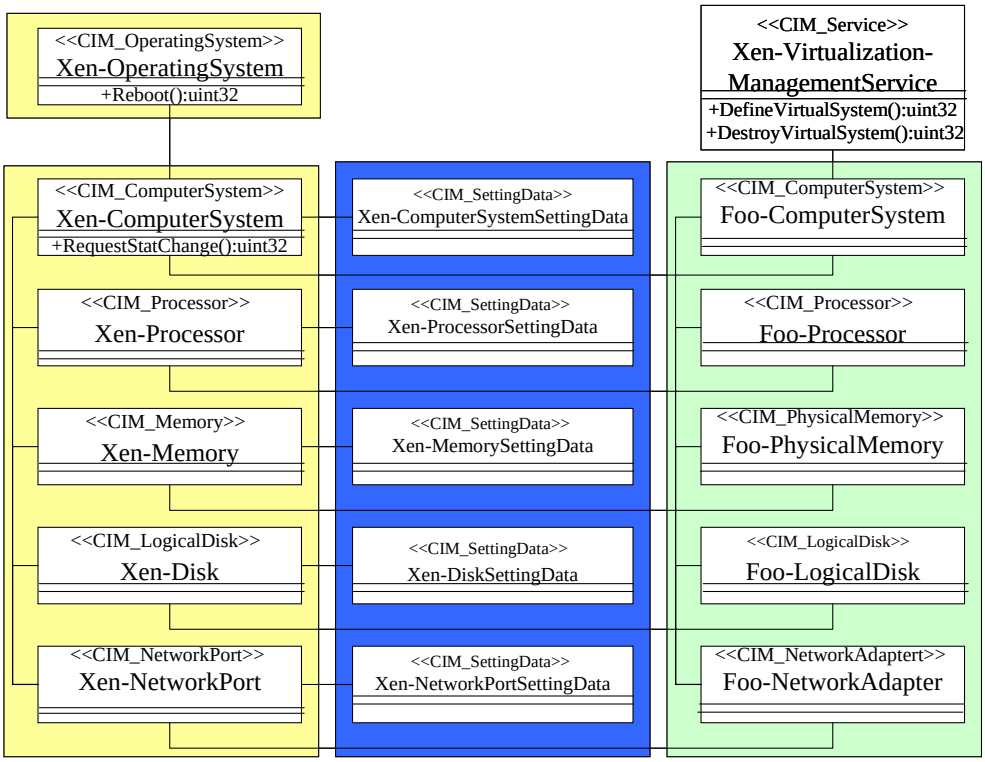

図 2.6 CIM による Xen のモデル

図 2.7に示すように、ホスト（Dom0）のCPU、メモリなどのリソース (ホストのLogicalDevice)は、ResourcePoolConfigurationServiceによりResourcePoolにプールされる[12]。ゲスト（DomU）の生成時に、このプールから適切なリソース (ゲストのLogicalDevice)が割り当てられ、割り当てについての情報がResourceAllocationSettingDataに保持される。

この際、仮想デバイスへのリソース割り当て（例 : Xen_Processor）と仮想デバイスそのもの（例 : Xen_ProcessorRASD（RASD=ResourceAlloacatorSettingData））は区別される。このようなモデルとすることにより、仮想化に対応していない CIM クライアントも、仮想化されたシステムを実際のシステムと同様に扱うことが可能になる。

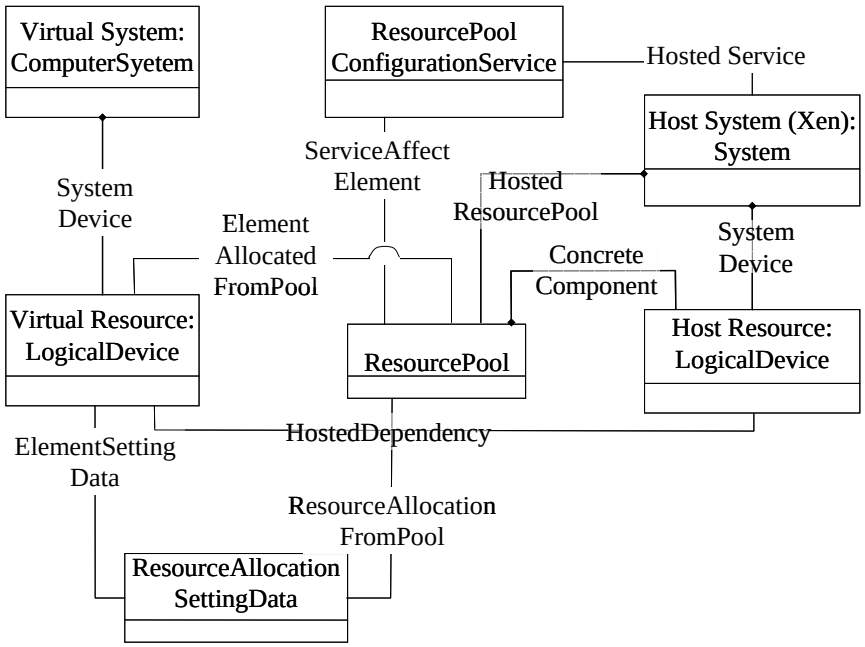

図 2.7　Xen-CIM におけるゲスト（DomU）のリソース割り当てとリソースプール

2.3.2　Web-Based Enterprise Management

Web-Based Enterprise Management (WBEM)は、CIM をベースに情報システムを運用管理するための一連の標準規格の集合体である。DMTF において、規格が策定されている。WBEM では CIM スキーマの XML での記述方法や、Web サービスから XML を介して CIM にアクセスする方法として、以下の 4 つの技術領域での標準規格を定義している。

(1)　マッピング

- WBEM URI Mapping Specification 1.0.0 (DMTF DSP0207)
- Representation of CIM in XML 2.2.0 (DMTF DSP0201)
- XML Document Type Definition (DTD) 2.2.0 (DMTF DSP0203)

(2)　プロトコル

- CIM Operations over HTTP 1.2.0 (DMTF DSP0200)
- Server Management Command Line Protocol Specification (SM CLP) 1.0.0 (DMTF DSP0214)
- WS-DistributedManagement (DMTF において WS-Management との統合を検討中)
- WS-Management (DMTF において WS-DistributedManagement との統合を検討中)

(3)　ディスカバリ

- WBEM Discovery using SLP (DMTF DSP0205)
- WBEM SLP Template (DMTF DSP0206)

(4)　検索言語

- CIM Query Language Specification 1.0.0 (DMTF DSP0202)

WBEMのアーキテクチャは図 2.8のようになっている。MappingsでCIMの内容を通信に適した形式に変換する方法を定義し、Protocolsで実際の通信で使用するプロトコルを定義し、DiscoveryでWBEMクライアントがWBEMサーバを検索する手段を定義し、Query Languageでリソース問合せの方法を定義している。なお、WBEMサーバが実際のソフトウェアやハードウェアと通信するためにCommon Manageablity Programming Interface (CMPI)が規定されているが、WBEMには含まれないために、ここでの説明は省略する。

WBEM をこのようなレイヤで定義することにより、ユーザインタフェースや、実際のソフトウェアやハードウェアから独立した標準規格となっており、汎用性がより高くなっている。

WBEM クライアントと WBEM サーバはオープンソースを含め、様々なものが開発されている。これらを利用することで、開発のコストを下げることが可能と考えられる。オープンソースとしては Open Group の OpenPegasus や OpenWBEM Project の OpenWBEM が挙げられる。

OpenWBEM は Novell が中心となって開発している BSD ライセンスのオープンソースである。SUSE Linux に実装されているが、2006 年 10 月 19 日に Ver. 3.2.2 がリリースされた後アップデートされていない。

OpenPegasus は Open Group で開発している MIT ライセンスのオープンソースである。パッケージ化されたものが RedHat Linux などでも配布されている。2007 年 10 月 19 日に Ver. 2.7 がリリースされている。

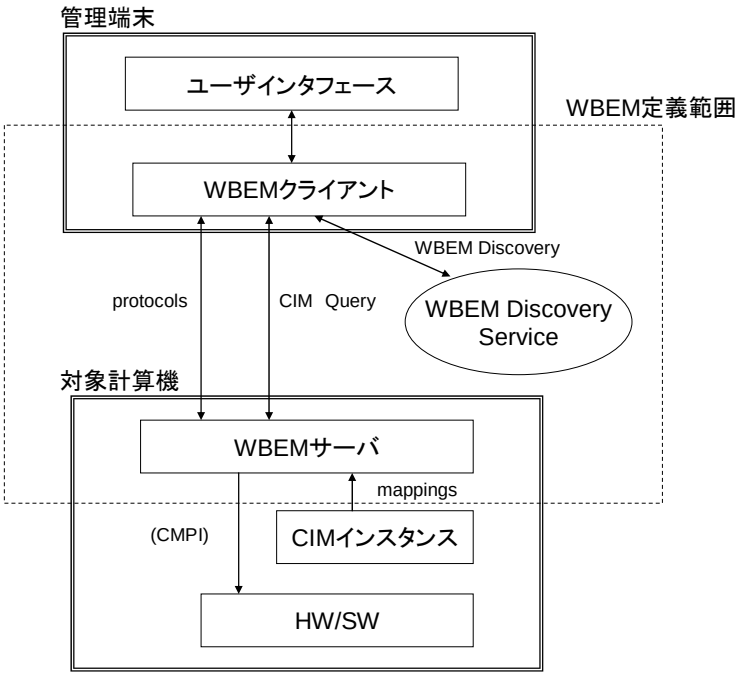

図 2.8　WBEM アーキテクチャ

以下、WBEM で定義されている各仕様の概要を記す。

(1) マッピング

(a) WBEM URI Mapping (CIM から URI へのマッピング)

WBEM において URI へのマッピングは、WBEM URI Mapping Specification 1.0.0 (DMTF DSP0207)で定義されている。この仕様では CIM 仕様で定義された CIM 名から URI へのマッピングを定義しており、WBEM URI は、CIM の各要素を特定するための文字列となる。WBEM URI は WBEM プロトコルが、CIM 要素をを特定するための手段として用いられる。

(b) xmlCIM (CIM から XML へのマッピング)

xmlCIM は、以下の２つの標準規格を用いて、CIM から XML へのマッピングを定義している。

- Representation of CIM in XML 2.2.0 (DMTF DSP0201)
- XML Document Type Definition (DTD) 2.2.0 (DMTF DSP0203)

xmlCIM は、CIM の要素を XML で表記できるようにデザインされている。xmlCIM は CIM の宣言の表記にも、CIM プロトコルで使用される CIM メッセージの表記にも使用できる。DMTF は xmlCIM に対する DTD(Document Type Definition)を CIM DTD として提供している。

(2) プロトコル

(a) CIM-XML プロトコル

CIM-XML プロトコルは、CIM で表記された情報を HTTP で送受信するためのプロトコルである。CIM Operations over HTTP 1.2.0 (DMTF DSP0200)に定義されている。CIM-XML は以下のコンポーネントからなる。

- CIM
- xmlCIM エンコード
- CIM データを扱うための操作方法の集合体
- HTTP カプセル化

CIM-XML では、CIM 情報を xmlCIM エンコード仕様で XML 形式にエンコードし、それを HTTP のペイロードとして扱う。CIM-XML は管理端末と管理インフラストラクチャ間の操作を、CIM メッセージのリクエスト・レスポンスの対として定義する。

(b) コマンドラインプロトコル

情報システムのリソースを運用管理するための共通的なコマンドの書式とコマンドを通信するためのメッセージについて、Server Management Command Line Protocol Specification (SM CLP) 1.0.0

(DMTF DSP0214)で定義されている。SM CLP は人間に解りやすい表記を用いたプロトコルである。CLP は以下について定義している。

- CIM スキーマのサブセットへのマッピング
- コマンドのシンタックスと文法
- 出力のフォーマット
- セッションプロトコル
- トランスポートプロトコル

SM CLP クライアントは MAP (Management Access Point)に Telnet や SSHv2 で接続し、テキストベースのプロトコルに基づいたリクエストを送信する。MAP では SM CLP サービスが動作し、SM CLP クライアントにレスポンスを返す。

(c) WS-DistributedManagement

WS-DistributedManagement (WSDM)は、2005 年に OASIS が承認した、分散リソースを Web サービスによって管理するための仕様である。下記の MS-Management と一部重複する部分もあるが、この仕様ではデータセンタなどでの大型のシステムにおける Web アプリケーションの管理が中心となっている。現在、HP、IBM、Intel、Microsoft が中心となって WS-DistributedManagement と WS-Management との融合が進められている。

(d) WS-Management

2006 年に AMD、DELL、Intel、Microsoft、Sun Microsystems が DMTF に提案した、分散リソースを Web サービスによって管理するための仕様である。上記の WS-DistributedManagement と一部重複する部分もあるが、この仕様では様々なデバイスや情報システムについての管理が中心となっている。現在、HP、IBM、Intel、Microsoft が中心となって WS-DistributedManagement と WS-Management との融合が進められている。

(3) ディスカバリ

(a) Service Location Protocol

WBEM Discovery は WBEM サーバの場所を発見するためのメカニズムである。WBEM はそのための仕様を定義しているが、これは将来別の Discovery メカニズムが付け加えられることを排除するものではない。Service Location Protocol (SLP)を使用した WBEM Discovery は、以下の 2 つの標準規格から定義される。

- WBEM Discovery using SLP (DMTF DSP0205)
- WBEM SLP Template (DMTF DSP0206)

SLP は IETF の RFC2608 で定義されている。SLP では SA (Service Agents)と表現されるサービス

の存在性、場所、コンフィギュレーションの情報に、クライアントがアクセスするためのフレームワークを定義している。SLP を利用すると、クライアントがサービス名やアクセス方法を知らなくても、クライアントはサービスのタイプを指定することで、必要なサービスにアクセスできる。

SLP ではリポジトリとして DA (Directory Agents)を置くことで、小規模から大規模までの環境に適応するスケーラビリティを確保している。WBEM Discovery に SLP を使用する際には、この DA も配置する必要がある。WBEM SLP Template では WBEM サーバに関する情報を定義している。

(4) 検索言語

(a) WQL/CQL

WQL (WBEM Query Language)は WBEM における検索言語の実装である。WQL は CIM に特化した言語である。この実装を仕様化し、DMTF の承認を得た標準が CQL (CIM Query Language)である。CIM Query Language Specification 1.0.0 (DMTF DSP0202) に定義されている。CIM インスタンスの集合から必要な情報を取り出すための検索式に使用できるキーワードには、以下のようなものがある。

表 2.1 CQL キーワード（一部）

キーワード	意味
SELECT	検索式のなかの属性の指定
WHERE	検索式の検索範囲の限定
FROM	SELECT 文のなかの属性を含むクラスを指定
NOT	比較のための演算子
OR	論理演算子
AND	論理演算子
LIKE	与えられた最小の情報量から結果を生成

典型的には、

“SELECT CIMinstance FROM CIMclass WHERE conditional_expression”

といった形式の検索式が用いられる。具体的な例としては、

“SELECT * FROM Solaris_FileSystem WHERE (Name = ‘home’ OR Name = ‘files’) AND AvailableSpace > 2000000 AND FileSystem = ‘Solaris’”

といった検索式が用いられる。

(5) 参照実装

WS-Management と関連仕様群は、さまざまな実装（商用製品、オープンソースソフトウェア）

で利用可能となっている。

WinRM（Windows Remote Management)は、WS-Management の Microsoft Windows 用実装である。WinRM は、Windows Vista、Windows Server 2008 には標準で含まれている。主に、ハードウェアの遠隔保守やモニタリングの用途を想定している。

Wiseman は、Glassfish プロジェクトにおける WS-Management の Java による実装である。WS-Management で定義されている機能のほとんど（WS-Management、WS-Management Catalog、WS-Addressing、WS-Enumeration、WS-Eventing、WS-Transfer）に対応している。

Openwsman は、インテルの Open-Source Technology Center が推進している、WS-Management の C による実装である。WS-Management で定義されている機能のほとんどを実装している。

2.3.3 CIM データベース

リソースの概念モデルである CIM を格納するためのデータベース、データベースにアクセスするための情報検索に関するインタフェースとして、種々の技術が応用されている。

(1) XML データベース

(a) 機能

CIM を格納するデータベースでは、XML データベースが応用される傾向にある。

①狭義の XML データベースは、XML のツリー構造をそのままデータ構造として保持するデータベースを指す。特に、ネイティブ XML データベースと呼ばれる。②最近では、リレーショナルデータベースに XML のツリー構造をマッピングして格納する方式や、単にテキストファイルとして XML を格納する方式など、様々なものがある。最近では、①、②を含め、XPath、XQuery で検索できるインタフェースを備えたデータベースを XML データベースと呼ぶことが多い。

リレーショナルデータベースは、一度作成されたデータ構造を運用中に変更することが困難である。一方、XML データベースは非常に拡張性が高い。XML の仕様が、スキーマを必須としておらず、途中でデータ構造が変化することを前提としたシステムを比較的容易に構築することができる。実用上の問題は、リレーショナルデータベースにおける SQL のような標準規格がないことであったが、XML データベースの検索に、XPath、XQuery の標準規格が策定され、製品においても採用が進んでいる。

(b) ソフトウェア

XML データベースとして、さまざまなソフトウェアが開発されている。XML データベースが研究開発フェーズから、本格的な普及期に以降していることが伺える。代表的なものを下記に列挙する。

表 2.2　XML データベース製品

名称	開発元	特徴
Shunsaku	富士通	XML データベースエンジン Interstage Shunsaku Data Manager と専用ハードウェア Shunsaku Engine。
TX1	東芝ソリューション	XML データベース。テラバイト級 XML データの高速検索に特長がある。
Cyber Luxeon	サーバーテック	オブジェクトデータベース ObjectStore をコアエンジンとした XML データベース。
NeoCoreXMS	アメリカ Xpriori	ネイティブ XML データベース。日本で販売数が多い。
EsTerra	メディアフュージョン	XML ストレージサーバ（DBMS）。
Yggdrasill	メディアフュージョン	XML データベース。機能を削減した GPL ライセンスのオープンソース版も提供されている。
Tamino	ドイツ SoftwareAG	XML データベース。世界での販売数が多い。
Xindice	オープンソース（Apache XML プロジェクト）	ネイティブ XML データベース。
eXist	オープンソース	Java で記述されたネイティブ XML データベース。

(c)　　インタフェース

CIMを格納するデータベースとしてXMLデータベースを使用する場合には、XMLデータベースへアクセスするインタフェースとして、汎用のXQuery [19]で検索式を記述することができる。XQueryは、XPath、XSLTをベースとする技術である。

XPath (XML Path Language)

XML に準拠した文書について、特定の部分を指定する言語構文である。XPath 自体は、XML に準拠したマークアップ言語ではない。 W3C (World Wide Web Consortium)で開発され、1999 年 11 月 16 日に、XML Path Language (XPath) 1.0 が、XSL Transformations (XSLT) 1.0 と同時に勧告として公表された。ハイパーリンクなどに使われる、人間が追加したアンカーによる位置の指定とは異なり、XML データを表すタグの木構造をたどって文書内のあらゆる要素や属性にアクセスする手段を提供する。

XSLT (Extensible Stylesheet Language Transformations)

XPath による選択と検索に基づき、XML 文書全体または文書の一部に対して変換を行い、別の XML 文書または表示・印刷用形式（XSL-FO、HTML、RTF、TeX 文書など）の文書を生成するための簡易言語である。XSLT による変換を実行するためのソフトウェアを「XSLT プロセッサ」

と呼び、XT や Xalan など様々なものが知られている。

XQuery

静的型付け機能（実装依存）を持つ XML データ問合せのための関数型言語である。XPath の目的は木の節点を指し示す(Addressing)ことであるが、XQuery の目的は XML データソースの照会機能を提供である。XQuery は、XML 文書における要素や属性を指定する XPath をベースとし、XPath 2.0 を含んでいる。XML 文書を格納するデータベースにアクセスすることが可能なだけでなく、データの比較、代入など、リレーショナルデータベースのように多様な問い合わせを行うことも可能である。

(2) 関連する技術

リソースの概念モデルである CIM を格納するデータベース、情報検索言語の他にも、リソースの構成を格納するための技術が開発されている。

(a) Meta-computing Directory Service

Meta-computing Directory Service (MDS)は、グリッドコンピューティングのためのオープンソースプロジェクトGlobus Alliance [21]のGlobusツールキットに含まれる情報検索サービスである。システムを構成する、静的・動的な情報へのアクセスを提供する。どのサイトで提供される情報にも同一の方法でアクセスできる、動的データへの効率的なアクセスがきる、サーバが分散した非集中型管理といった特徴がある。

エージェント機能の GRIS (Grid Resource Information Service)と、マネージャ機能の GIIS (Grid Index Information Service)を階層的に積み上げて構成される。データの格納には、LDAP (Lightweight Directory Access Protocol)を使用している。

2.3.4 ITIL ベースの構成管理

ITILv3 に従う構成管理の技術として CMDBf がある。CMDBf（CMDB Federation）は、2006 年に発足した業界団体である。構成管理データベース（CMDB）と管理データリポジトリ（MDR）との間で、データの検索やエクスポートを行う仕様として、CMDB Federation version 1.0 が公開された。SOAP, WSDL, XML Schema といったウェブサービス仕様を利用し、管理データを連携させる。検索サービスを通して、CMDB から MDR の管理データにアクセスできる。

IBM Tivoli CCMDB は、ITIL で必要とされる CMDB の機能を含む仮想的 CMDB で、ITILv3 に対応している。変更管理と合わせた実装である。変更内容が承認されたものか判別する機能を持つ。レポート作成やインパクト分析には、他の製品と組み合わせる。

HP Universal CMDB は、ITILv3 の要件を満たす CMDB の基本機能のほかに、インパクト分析、アクセス制御、管理サービスの機能も提供する。

CA Unicenter CMDB は、ITIL に対応する構成管理製品群に含まれる CMDB である。この製品

群によって収集される資産情報を CMDB の構築に利用している。

2.3.5 まとめ

リソース管理に関連する技術動向の要点を以下にまとめる。

- リソース管理は、ハードウェア・ソフトウェアなどのリソースの識別子、属性、相互の関係などの情報を統一的なデータベースに保持し、更新前後の情報システムの状態の把握やリソースに関する情報の収集・検索に使用される。これにより、コスト削減、迅速な情報システムの構成変更、相互運用性の利便性改善、セキュリティ設定の誤りの発見などの効果が期待できる。
- リソース管理に関する中心的な標準規格として、DMTF で CIM と WBEM が開発されている。CIM の仮想マシンモニタへの拡張や、WBEM にもとづく情報システムの運用管理全般の標準化などが推進されており、Xen-CIM や OpenPegasus などのオープンソースの実装の開発も進んでいる。また、XML をベースとした検索技術などの新しい技術の開発が進んでいる。
- WS-Management と関連仕様群は、WinRM（Windows Remote Management)、Wiseman、Openwsman といったさまざまな商用製品、オープンソースソフトウェアで利用可能となっている。
- ITILv3 に従う構成管理の技術として CMDBf（CMDB Federation）がある。構成管理データベース（CMDB）と管理データリポジトリ（MDR）との間で、データの検索やエクスポートを行う仕様として、CMDB Federation version 1.0 が公開された。IBM Tivoli CCMDB、HP Universal CMDB、CA Unicenter CMDB といった多くの商用製品が、ITILv3 の要件を満たす CMDB を提供している。

2.4 ポリシ適用

一般に、セキュリティポリシが規定するアクセス権は、サブジェクト(S)、オブジェクト(O)、アクション(A)の組(S,O,A)に対する可否であると定義できる。可否を管理・実施する機能が、アクセス制御ポリシ適用である。

アクセス制御ポリシ適用には、大きく 2 つの技術的側面がある。第一の側面は、セキュリティポリシの記述方法である。記述方法には、セキュリティポリシの意味、規約、管理方法などを表すアクセス制御ポリシモデルと、具体的な表現としてシンタックスを規定するアクセス制御ポリシ言語とがある。第二の側面は、セキュリティポリシを実施し、アクセスの可否を制御するアクセス制御ポリシ適用機構である。広義には、アクセス制御ポリシ適用機構には、セキュリティポリシの実施だけでなく、セキュリティポリシの生成、配布の機能も含まれる。以後、誤解の生じない範囲で、アクセス制御ポリシ適用を単にポリシ適用、アクセス制御ポリシモデルを単にポリシモデルと呼称する。

情報システムが、ポリシ適用を導入すると、さまざまな利点が得られる。

① ポリシ適用機構は、コストを削減する。大規模な情報システムには、多数のユーザや部門が関係しており、数多くのセキュリティポリシの設定が必要になる。セキュリティポリシを棚卸しする作業を集中化・自動化することで、設定ミスをなくし、人手による労力を削減することができる。

② ポリシ適用機構は、迅速な情報システムの構成変更を可能にする。新規のサービスの立ち上げや新しいユーザの権限の変更を行う際に、既存のインフラストラクチャをもとに、容易にセキュリティポリシを適用できる。ポリシ適用を人手で確認する必要が削減され、確認やテストに要する時間を短縮することができる。

③ ポリシ適用機構によって、セキュリティポリシを一元的に集中管理することで、機器にまたがるセキュリティの確保が可能になる。

④ ポリシ適用機構によって、設定ミスやセキュリティポリシの適用し忘れを回避することができる。

2.4.1 アクセス制御ポリシモデル

生成（create）、読込（read）、書込（write）、および消去（destroy）の 4 種のアクセス権を制御する代表的な管理モデルとしては、任意アクセス制御モデル(DAC: Discretionary Access Control)と強制アクセス制御モデル(MAC: Mandatory Access Control)に分けられる。DAC モデルは、オブジェクトの所有者がアクセス権の設定を自由に設定できるモデルであり、機密情報などの管理用途には向いていない。このため、政府や軍事系のシステムでは、機密情報の取り扱いを可能とするため、情報の所有者であっても自由にアクセス権の設定を許さないアクセス制御モデルとして MAC モデルが使われるようになった。

一方、ロールベースアクセス制御モデル(RBAC: Role Based Access Control)は、ポリシ中立のモデルといわれており、DAC、MAC の両方に適用可能なモデルとなっている。本章では、セキュア OS で主流となっている、RBAC、MAC の各モデルを概説するとともに、アクセス制御の最近の研究の中心となっている、職務分掌などの制約条件のモデルに関して概説する。

(1) RBAC 基本モデル

RBAC の特徴は、ユーザに役割（ロール）を割り当ててから、役割に対してパーミションを定義する点である。

標準的なRBACモデルとしては、米国の国立標準技術研究所(NIST: National Institute of Standards and Technology)がCore-RBACモデルを定めている[44]。そのモデルを図 2.9に示す。

Core-RBAC モデルでは、ユーザ(USER)へのロール(ROLES)の割り当ての他に、ユーザの一時的な状態(SESSIONS)を定義しており、同時に複数の異なったロール(ROLES)を所有できるようになっている。SESSIONS の概念がないモデルは Flat RBAC と呼ばれている。

Core-RBAC モデルでは、ユーザもしくはセッションに割り当てられたロールに対して、パーミ

ションが割り当てられる。ここで、パーミションはオブジェクト(アクセスの対象物)とオペレーション(オブジェクトにアクセスを行う主体、一般的にはプロセスなど)の関係を示したものであり、例えば、読み出しが可能などの動作を許可するものである。パーミションは、それが割り当てられたロールにおいてで、プロセスからオブジェクトに対するアクセス可否を定義する。

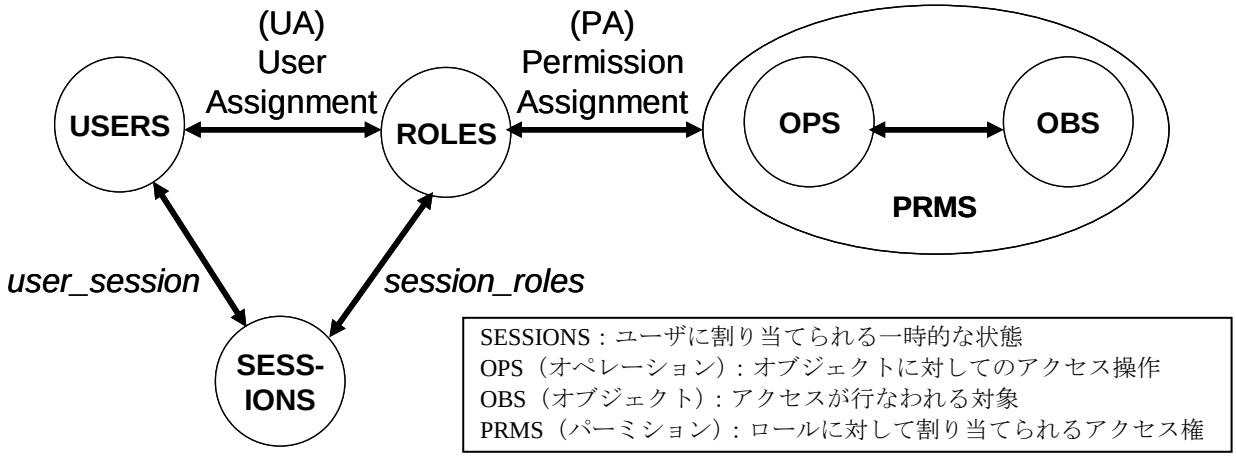

図 2.9 NIST の Core-RBAC モデル

(2)　　強制アクセス制御モデル

強制アクセス制御モデルには、情報のセキュリティレベルにより区分けを行う階層化セキュリティモデルと、情報の種類に応じたアクセス制御を行う区分型セキュリティモデルがある。さらに、階層化セキュリティモデルには、秘匿性モデルと完全性モデルがあるが、これらについては 4.2 節を参照されたい。

本章では、階層化セキュリティモデルの一種であるが、情報のフロー制御を一般化した束モデル(Lattice モデル) **エラー! 参照元が見つかりません。**に関して概説する。

束モデルは、情報を完全に区分けするのではなく、情報のフローに応じて、情報へのアクセス制御を変化させていくモデルであり､階層化セキュリティモデルのベースとなったモデルである。

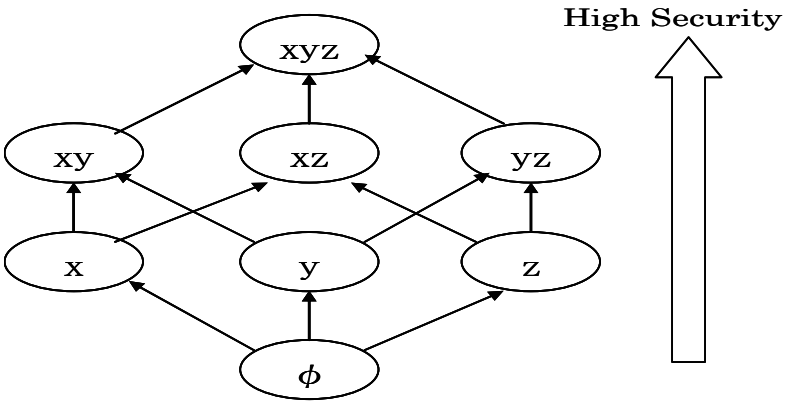

図 2.10　Lattice モデル

このモデルでは、セキュリティレベルはφが低く、χ={x,y,z}が高くなっている。セキュリティレベルとして包含されている方向(高い方向)にのみ情報が流れるモデルであり、情報の秘匿の観点から安全性が証明されているモデルとなっている。図 2.10では、“ｙ”の情報は“ｙ”を包含する関係を持つ、“ｘｙ”または“ｙｚ”にしか流れないというモデルを定義している。

(3)　　職務分掌と RBAC 拡張モデル

RBAC の拡張モデルとしては、NIST がロール階層化とロール割り当ての制約を付加したモデルを定義している。

拡張モデルでは、Core-RBAC モデルに、ロールの階層化を行ったモデルと、 ロールの割り当てに制約を付けた 2 つのモデルを定義している。

(a)　　階層化 RBAC(Hierarchal RBAC)

ロールに階層化構造を持たせたモデル。組織の階層構成(役職など)に合わせた権限の階層化が可能となる。基本的な階層化の考え方としては、上位の階層のロールは、そのロールと接続されている下位のロールの権限を有するといった制御が可能となる。

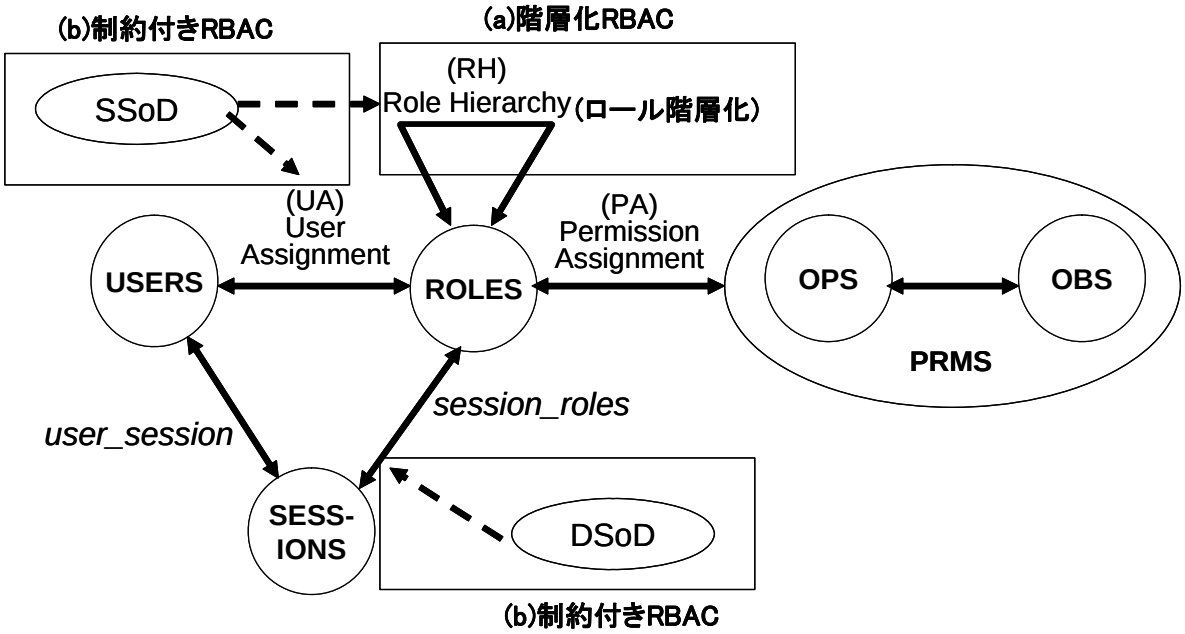

図 2.11　拡張された RBAC モデル

(b)　　制約付き RBAC(Constrained RBAC)

ロールの割り当ての制約を持たせたモデル。色々な制約を付けることが可能であるが、NIST の拡張モデルでは、制約を静的制約と動的制約との 2 種類に分類している。

- 静的制約条件
  静的制約とは、ロールを設定する際に、ロールに課す制約条件を指す。ユーザへロールを割り当てる場合やロールの階層を設定する場合に、必ず満たさなければならない。例えば、「同一のユーザに、相反するロールＡとロールＢを割り当ててはいけない」といった制約を規定する。
- 動的制約条件
  動的制約とは、ロールを設定する際に、一時的なセッションに課す制約条件を指す。そのセッションにおいて、一時的な権限を設定する場合には、ユーザに割り当てるロールには制約が加えられる。例えば、「現在ロールＣを設定されたユーザには、同時にロールＤを設定することができない」といった制約を規定する。

ロール階層化とロール割り当ての制約を付加した RBAC は、最近の研究動向として重要視されており、職務分掌(SoD: Segregation of Duties)の実現に有効なモデルである。職務分掌とは、企業や組織において、権限や責任の範囲を定義したものであり、内部統制には重要な事項となっている。よく用いられる例としては、銀行振込書の作成者と、承認は別人でなければならないといっ

た、排他的な制御が求められている。NIST の拡張モデルでも、静的制約条件と動的制約条件とに対応して、静的職務分掌(SSoD : Static Seperation of Duty)と動的職務分掌(DSoD : Dynamic Separation of Duty)とを定義している。

(4)　　アクセス制御ポリシモデルに関する最近の研究動向

アクセス制御ポリシモデルの最近の研究動向としては、上記の職務分掌に対応する研究の他に、以下のような研究が行われている。

- ロール管理体系に RBAC の概念を導入したモデル
  ロールの管理体系の中に管理ロールという概念を導入し、さらに管理ロールにも階層を持たせることにより、管理ロールに従った範囲でのロール管理だけをできるようにしたARBACモデル**エラー! 参照元が見つかりません。**[45]が提案されている。
- 大規模組織に対応した管理を実現するモデル
  実際に大企業でRBACを実現しようとした場合には、RBACを用いた場合でも、管理が複雑となり、実際の運用に困難さをともなう。管理の複雑さを軽減するためのモデルが研究されており、組織レベルの管理とシステムレベルの管理を分離したモデル[46]、抽象的なレベルと具体的なレベルに分離して管理することを特徴としたOrBAC(Organization Based Access Control)モデル[47]などが提案されている。

2.4.2　アクセス制御ポリシ言語

アクセス制御ポリシを記述する代表的なポリシ言語を 2 つ紹介する。

(1)　　XACML

SELinux向けポリシ言語 [48] では、SELinuxが提供するアクセス制御機構に特化した設計となっていた。多くのポリシ言語が、OS・機器に固有の設定となっていた。したがって、多様なOS・機器を含むヘテロな情報システムで求められる相互運用性が不十分であった。この問題に対し、XACML (Extensible Access Control Markup Language) [49] が 2003 年にOASISにより策定された。XACMLはXMLに基づく言語であり、拡張性と可読性を高めることを目標に仕様が設計されている。図 2.12に示すように、XACMLでは、アクセス制御ポリシの基礎モデルであるサブジェクト（<Subject>）、オブジェクト（<Resource>）、アクション（<Action>）を記述するタグが定義されている。また、サブジェクトやオブジェクトに対する属性条件を記述するタグ（<SubjectAttributeMatch>や<ResourceAttributeMatch>）が定義されている。これにより、サブジェクトのロールやオブジェクトのセキュリティラベルを記述できる。

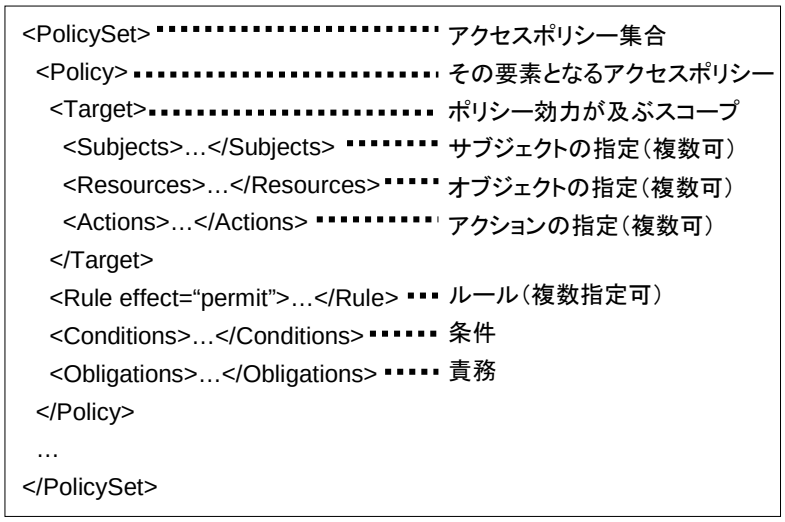

図 2.12　XACML の構文

(2)　RBAC Profile

XACML RBAC Profile [50] とは、XACMLを用いてRBACモデルのアクセス制御ポリシを記述する際の拡張規格である。RBACモデルを記述するための以下のようなタグ、属性、要素などが定義されている。

- ポリシ記述中でのロール ID の指定方法（例：&role;manager, &role;employee）
- ロールと権限のマッピング方法（例：<PolicySet PolicySetId= "&role;manager">）
- ロール階層の定義方法（<PolicySetIdReference>を用いて、親ロールの権限を参照する方法）

2.4.3　アクセス制御ポリシ適用機構

近年、内部統制の対応強化の流れの中で、企業を中心に、ID 管理の統合と共に統合的なポリシ適用システムの重要性が認識されつつある。特に、COBIT フレームワークが求めるように、RBAC に対するポリシ適用機構が求められている。簡単のため、ここでは RBAC モデルのアクセス制御ポリシを RBAC ポリシと呼ぶことにする。RBAC ポリシを実施し、アクセス制御を行う仕組みを RBAC ポリシ適用機構と呼ぶことにする。

ポリシ適用機構の中核的な仕組みが、リファレンスモニタである。リファレンスモニタとは、コンピュータ上に存在するすべてのデータに対するアクセスの可否を制御する仕組みを指す。任意のサブジェクトからオブジェクトへのすべてのアクセスを仲裁する仕組みということもできる。リファレンスモニタの機能要件として、すべてのアクセスを検出できること、悪意のあるプロセスによってアクセス制御を邪魔されないこと、アクセス制御の結果を検証できることが必要である。

以下では、企業 IT システムの構成要素である VM、OS、アプリケーション、ネットワークに対する RBAC ポリシ適用機構を概観する。

(1) VM へのポリシ適用

現在、VMware、Xen、Windows Virtualization など多様な VMM の開発に見られるように、VMM の黎明期にあるため、ポリシ適用を統合する動きはまだ見られず、RBAC 等のポリシ強制機能をもつリファレンスモニタの開発が中心である。一例として、Xen のリファレンスモニタである sHype を紹介する。

sHype[51]は、管理OS上で稼動するポリシマネージャと、Xen VMM内部に組み込まれたリファレンスモニタで構成され、ゲストOS間の情報フロー（IP通信やディスク共有など）に対するRBACポリシ適用を実現している（図 2.13）。

libvirt は仮想マシンの制御を抽象化したライブラリである。レッドハット社を中心としたオープンソースの開発プロジェクトである。libvert は、Xen、KVM、QEMU などをサポートしている。セキュリティ拡張として、libvert に MAC (Mandatory Access Control)アクセス制御ポリシモデルを統合するためのオープンソースの開発プロジェクトが sVirt である。

また、CAのAccess Control[1]では、管理OSにおける管理コマンド実行を制御することにより、従来絶対権限を持っていたVM管理者の権限最小化を実現している。VMware ESX Server・Microsoft Virtual Serverなどの仮想化ソフトに加えて、メインフレームz/VM・HP-UX LPARなど、幅広い仮想化機構に対応している。

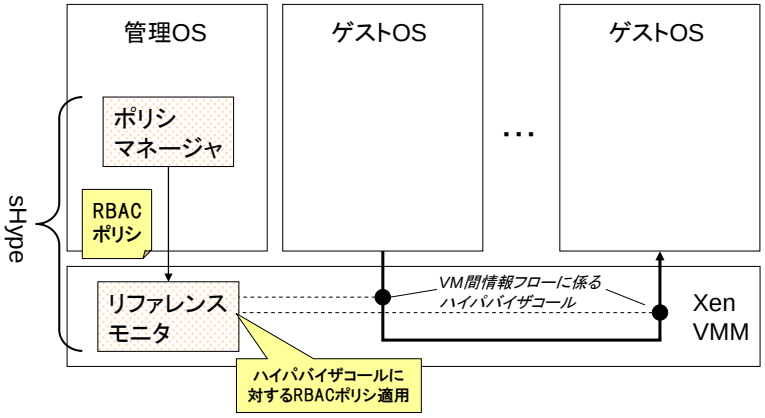

図 2.13 sHype のアーキテクチャ

(2) OS へのポリシ適用

RBAC ポリシをサポートする OS として、SELinux がよく知られている。通常、SELinux では一連のコマンドラインツールでポリシを生成・適用するが、設定不備が起こりやすく、扱いが難し

[1] 従来eTrust Access Controlという名称であったが、Access Control に改称された。

いとされており、近年ポリシ生成・適用の容易化に関する技術開発が行われている。一例として、SEEdit を紹介する。

SEEdit[52]は、オープンソースのSELinuxポリシエディタであり、GUIによるポリシ生成機能や、SEEdit独自の簡易ポリシ記述言語であるSPDL(Simple Policy Description Language)からSELinuxポリシ記述への変換機能などにより、SELinuxへのポリシ適用の容易化を実現している（図 2.14）。

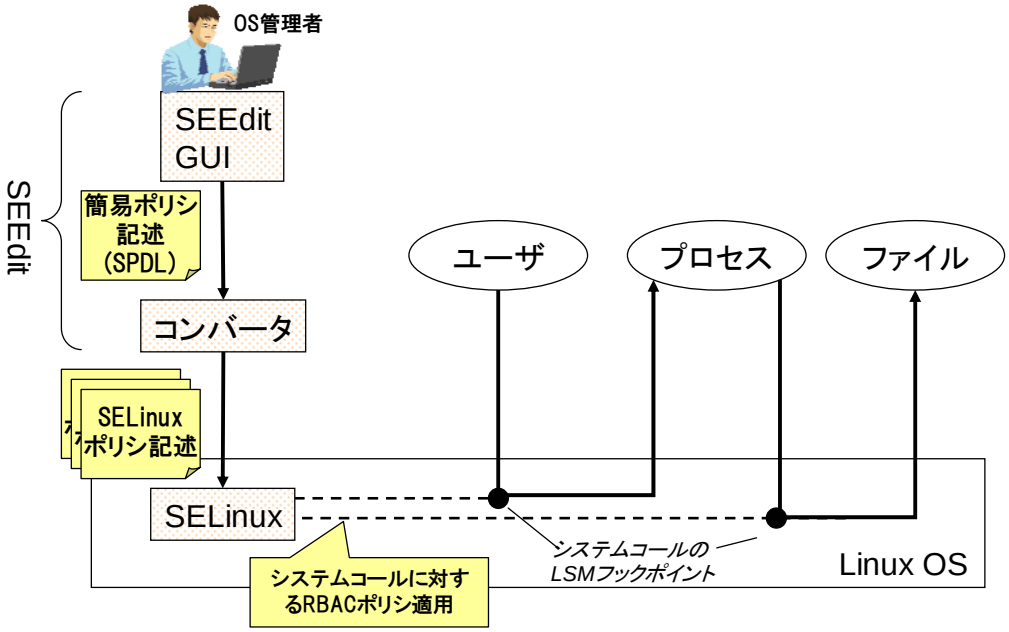

図 2.14　SEEdit における SELinux ポリシ適用方式

また、CA の Access Control は、特権ユーザも一般ユーザのように扱って OS 管理者権限を最小化する。具体的には、ファイル、ディレクトリ、TCP サービス、プロセス、su コマンドといったリソースへのアクセスを制御する。また、すべてのセキュリティに関わるイベントを監視・追跡できる監査機能をもつ。異なる OS でも、同じ形式でアクセスログを取得できるため、セキュリティ管理に必要なレポートを一元的に作成できる点が特徴とされる。

VM 環境への適用については、前述の管理 OS への適用のほか、UNIX (Solaris, AIX, HP-UX)、Microsoft Windows、RedHat Linux などのゲスト OS にも適用して、ゲスト OS 管理者権限も最小化できる。一元化されたポリシ管理システムからターゲット OS への自動的なポリシ配布・適用を行うことができる。

(3)　　アプリケーションへのポリシ適用

現在の業務アプリケーションは開発が容易な Web アプリケーションとして実装されることが

多いが、近年、アプリケーションサーバの乱立による保守コスト増加が問題視されるようになった。この問題は、ポリシ管理・適用でも同様であり、アプリケーションサーバへのアクセス制御を行うゲートウェイを配置し、ID 管理機構と連携させることで、水平統合的な RBAC ポリシ適用を行うアプローチが主流となっている。一例として、Sun Microsystems から製品化されている Sun Java System Identity Manager と Sun Java System Access Manager を紹介する。

Sun Java System Identity Manager [53]はID管理製品であり、Sun Java System Access Manager[54]（PDP・PEP）に対するPAPとして動作する。ここで、PDP (policy decision point)とは、ポリシにもとづいてアクセス可否の判断を行う主体を指す。主体は、オペレータ、ソフトウェア、ハードウェアいずれの場合もある。PEP (policy enforcement point)とは、PDPの判断にしたがって、アクセス制御を実施してポリシを適用する主体を指す。PAP (policy administration point)とは、ポリシの内容や記述を生成する主体を指す。この連携ソリューションにおいて、Sun Java System Identity Managerは（１）RBACポリシの作成・配付機能や、（２）所定の業務手順とアクセス履歴との比較などの監査ポリシ検証機能などを実現している。

本ソリューションは、Webベースの業務アプリケーションを対象としており、図 2.15に示すような構成をもつ。セキュリティ管理者は、Sun Java System Identity Managerを用いて、ユーザID情報・ロール情報・RBACポリシを生成し、所定のLDAPリポジトリ（PIPに相当）に配付する。PIP (policy information point)とは、ポリシを適用する情報システムの属性を提供する主体を指す。例えば、ポリシ記述が参照するIDやリソースに関する情報を提供するID管理やリソース管理は、ポリシの観点からはPIPの一種である。Sun Java System Access Managerは、クライアントから業務アプリケーションサーバへのアクセスに対するゲートウェイとして動作し、LDAPリポジトリ内のID情報に基づくユーザ認証と、ロール情報・ポリシに基づいてアクセス制御やアクセスログの収集などを行う。

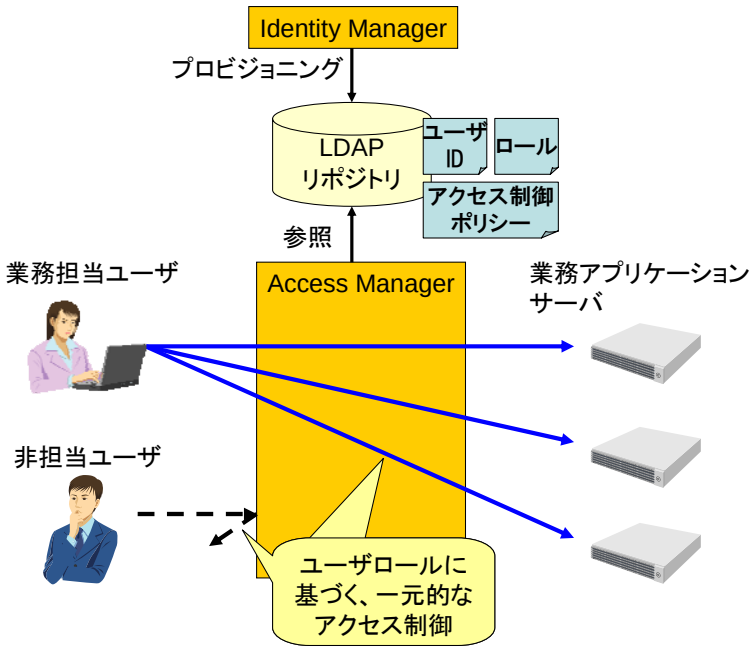

図 2.15　Sun Java System Identity Manger・Access Manager

(4)　　ネットワークへのポリシ適用

近年、NAC（Network Access Control）と呼ばれる、統合 ID 管理と連携するネットワークアクセス制御機構（スイッチやファイアウォール等）の開発が盛んである。NAC は従来の IP フィルタリングルールのようなユーザが実行可能（不可能）なアクションを記述するアクセス制御リスト(ACL)に代わり、ロールに基づく RBAC ポリシの適用を可能にする。一例として、Cisco Systems から製品化されている Cisco Application Control Engine を紹介する。

Cisco Application Control Engine (Cisco ACE) [55]は、ネットワークアプリケーションに対するRBACをサポートしたL4-L7 スイッチである。本製品は、エンドユーザのロールに応じたRBACポリシの適用により、ネットワークの論理的なパーティショニング（VLAN）を実現している（図 2.16）。

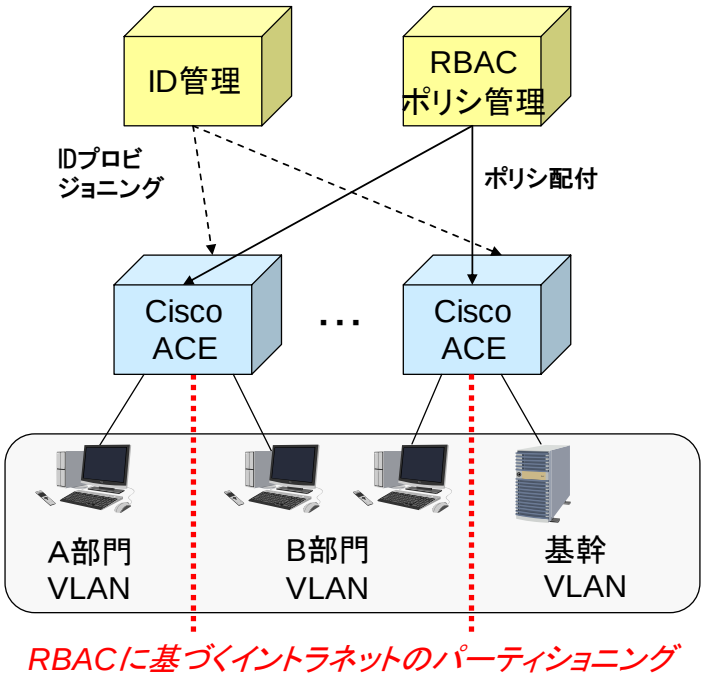

図 2.16　Cisco ACE の 1 ユースケース

2.4.4　リソース使用量制限ポリシ適用

OS には、リソース使用量の制限を設定することで、ユーザやプロセスが占有するリソース使用量を制御する機能を持つものがある。ポリシで規定されたファイル使用量、メモリ使用量、CPU 使用量、NW 使用量の上限を守るよう、リソースの割り当て制御を行う。それにより、プロセスや仮想マシンの間の性能干渉を避けたり、サービスレベルを保証したりする用途に応用できる。

リソース使用量制限についてのポリシによって規定された内容を適用する技術に関して、実装技術・対応製品の動向について述べる。

(1)　Linux

Linux には、ファイル使用量制御、メモリ使用量制御、CPU 使用量制御、NW 使用量制御のためのサポートツールが開発されている。表にまとめる。

表 2.3　Linux のリソース使用量制限

	ファイル使用量制限	メモリ使用量制限	CPU 使用量制限	NW 使用量制限
setrlimit	○	○	○	×
CKRM	×	○	○	×
CABI	×	○	○	×

setrlimit を呼び出したプロセス、あるいは、そのプロセスを実行したユーザごとに、使用できるリソース使用量の上限値を設定するシステムコールである。Linux カーネルは、設定したリソース使用量の上限値とプロセスが消費しているリソース使用量を比較し、プロセスがリソース使用量の上限値を超えて要求してもリソースを割り当てない。setrlimit は、ファイル使用量制御、メモリ使用量制御、CPU 使用量制御を行うことができる。

CKRM(Class-based Kernel Resource Management)は、ユーザが階層的なクラスを設定し、クラスごと使用できるリソース使用量の制御を行う仕組みである。CKRM は、元来、SUSE LINUX と IBM が共同開発したカーネルリソース管理システムである。カーネルリソースを動的に割り当てることが可能で、1 つのプロセスが CPU などのリソースを占有することを防止できる。仮想マシン間の性能干渉を避けたり、サービスレベルを保証したりする用途が考えられている。カーネルとユーザ空間で動作するアプリケーションとして実装されている。CKRM は、メモリ使用量制御、CPU 使用量制御を行うことができる。

CABI(CPU Accounting and Blocking Interfaces)は、複数のプロセスを一つの AO (Accounting Object)にバインドして、AO ごとに CPU 使用率を制御する仕組みである。早稲田大学で開発されており、リネオソリューションズ社とトライピークス社から公開されている。CKRM は、メモリ使用量制御、CPU 使用量制御を行うことができる。

(2) Windows

Windows には、ファイル使用量制御、メモリ使用量制御、CPU 使用量制御、NW 使用量制御のためのサポートツールが開発されている。表にまとめる。

表 2.4 Linux のリソース使用量制限

	ファイル使用量制限	メモリ使用量制限	CPU 使用量制限	NW 使用量制限
BES	×	×	○	×
Win32API	×	○	○	×
System Service	○	○	○	?

BES（Battle Encoder Shirase）は、プロセスごとに CPU 使用率の上限を設定できる GPL ライセンスのフリーソフトウェアである。プロセスが利用する CPU のタイムスライスを調整して実現している。BES は、CPU 使用量制御を行うことができる。

Win32API の一部として公開されている、Windows カーネル内での制御を行うための SetInformationJobObject 関数、SetProcessWorkingSetSize 関数を用いて、リソース使用量の上限を設定することができる。SetInformationJobObject 関数は、プロセス集合に対するリソース使用量の上限値を設定できる。SetWorkingSetSize 関数は、プロセスに対する物理メモリ容量の上限値を設定

できる。Win32API は、メモリ使用量制御、CPU 使用量制御を行うことができる。

System Service(システムコール)の ZwSetInformationJobObject 関数、ZwSetInformationProcess 関数、ZwSetQuotaInformationFile 関数を用いて、リソース使用量の上限を設定することができる。ZwSetInformationJobObject 関数は、プロセス集合に対するリソース使用量の上限値を設定できる。ZwSetInformationProcess 関数は、プロセスに対する物理メモリ容量の上限値を設定できる。ZwSetQuotaInfomationFile 関数は、ユーザに対するファイル使用量の上限値を設定できる。System Service は、ファイル使用量制御、メモリ使用量制御、CPU 使用量制御を行うことができる。

2.4.5 まとめ

ポリシ適用に関連する技術動向の要点を以下にまとめる。

- ポリシ適用機構により、セキュリティポリシを棚卸しする作業の集中化・自動化によるコスト削減、人手による確認やテストをなくし、新規のサービスの立ち上げや新しいユーザの権限の設定を迅速化、一元的な集中管理による機器にまたがるセキュリティの確保、設定ミスやセキュリティポリシ未適用の回避などの効果が期待できる。
- ポリシ適用には、①ポリシの意味、規約、管理方法などを表すポリシモデル、②シンタックスを規定するポリシ言語、③生成、配布したポリシを実施しアクセスの可否を制御するポリシ適用機構の 3 つの技術的な側面がある。XACML の RBAC ポリシへの拡張、VM や OS において RBAC ポリシを実施するためのリファレンスモニタの開発が進んでいる。ロール階層化とロール割り当ての制約を付加した RBAC は、職務分掌を実現するためのモデルとして注目されている。
- OS には、リソース使用量の制限を設定することで、ユーザやプロセスが占有するリソース使用量を制御する機能を持つものがある。ポリシで規定されたファイル使用量、メモリ使用量、CPU 使用量、NW 使用量の上限を守るよう、リソースの割り当て制御を行う。それにより、プロセスや仮想マシンの間の性能干渉を避けたり、サービスレベルを保証したりする用途に応用できる。Linux では、setrlimit、CKRM、CABI などが使用できる。Windows では、BES、Win32API、System Service などが使用できる。

2.5 ログ管理（監査）

本節では、IT システムの内部統制・監査を円滑に行うための技術について述べる。特に、現在ニーズが急速に高まっているログ管理と、そのフォレンジック対応に注目する。

2.5.1 監査とログ管理

監査報告に記述される監査人の意見は、十分な量と適切な質の監査証拠に基づいて作成される。監査人は、情報システムに対して監査する技術を十分に習得していても、情報システムにおける監査証拠が正しいものか、偽りのものかを鑑定する専門家ではない。そこで、通常、監査人が判

断の根拠とするものは、電子文書の作成、保管に関する統制状態である。電子文書の作成、保管に関する統制状態には、①監査証拠が原本か・複製か、②文書化されているか・されていないか、③監査対象組織の内部または関連組織で作成されたものか・第三者で作成されたものか、④監査人が直接取得できたものか・提供されたものか、などがある。このような統制状態を証拠力と呼ぶ。証拠力とは、監査人が監査証拠は正しいと判断する根拠の強さを指す。証拠力が強いものは、それぞれ「原本」、「文書化されている」、「第三者で作成されたもの」、「監査人が直接取得できたもの」である。監査人が直接取得する例としては、IT システムに対して実際に運用テストを行い、情報システムへの入力と出力および処理履歴の整合性記述した記録を確認することによって得られる監査証拠がある。

監査証拠としてアプリケーションやプラットフォームから出力されるログは、大変重要である。ログ内容の信頼性を高めるためには、前述の証拠力が強いログが必要である。そのためには、①フィルタリングされたログや統計値として加工されたログよりもアプリケーションやプラットフォームから直接出力された生ログ、②適切な内容を持つ文書化されたログ、③ひとつのアプリケーション、ひとつのプラットフォーム要素から個別に出力されたログの塊でなく、組織で利用しているアプリケーション、プラットフォームから出力されるログを一元管理できる統合ログ、④統合ログから業務・トランザクション・データベースの関連した動きを追跡できるログ解析システム、が求められる。

ログ内容に関する要件は、4W1H、すなわち、Who（誰が）、When（いつ）、Where（どこで）、What（何を）、How（どうした）が全て記述されていることである。また、記述内容は、大きな粒度ではなく、より小さな粒度の正確な記述が必要である。現在、多くのログで記述されている内容を以下に述べる。

(1) Who

「誰が」を識別するものとして、ユーザ名、ホスト名、端末名、IP アドレス、回線識別子、アプリケーション名、プロセス名がある。付加できる情報として個人認証ＩＣカード識別子や TPM に登録されたハードウェアの識別子があげられる。正しいログであるためには、ログに出力された「誰が」が特定の ID と 1 対 1 の対応関係にあること、その ID の利用に関して有効な個人認証が実施できることが必要である。

(2) When

「いつ」は、システム時刻を記述する。特に複数の情報システムのログを統合的に追跡する場合、組織内で正確な同じ時刻を持つことが要求され、一般には NTP サーバを利用する。また、重要性が高いログに関しては、公的認証を受けた時刻認証（タイムスタンプサービス）を利用し、正確な時刻の証拠の作成と同時にログの改ざん検知の対策を実施する。

(3) Where

「どこで」は、処理された場所でホスト名、OS タスク名、アプリケーション名などのログの出力元を記述する。

(4)　What

「何を」は、アクセス対象、操作対象の OS、OS コマンド、ファイル、DB、DB テーブル、プリンタや通信ソケットなどのリソースの名前、識別名、識別番号などを記述する。

(5)　How

「どうした」は、ログイン成功、アクセス拒否、故障発生、ファイル更新、印刷開始、不正パケット受信などの処理とその結果や、アプリケーションの処理内容とその結果を記述する。

なお、ユーザの認証情報（アカウントのパスワード等）やシステムが使用する秘密鍵などのセキュリティ上重要な情報は、たとえ暗号化していようともログに出力してはならない。

ログの信頼性を高めるための措置としては、①出力されるメッセージ内容を事前に規定する文書化、②出力されたメッセージ内容の正しさを検証した記録の保存、③出力された時刻の正しさを検証した記録の保存、④出力されたログの改ざん検知・防止や紛失防止、⑤出力された全てのログを一定期間保管する管理、⑥適切な保管ログのアクセス管理と秘匿化、等がある。メッセージ内容の正しさは、システム開発時、仕様変更時および監査時の運用テストによって、アプリケーションやプラットフォームの処理とそこから出力されたログの関係が正しいか判定し、文書として記録保存しておくことが必要である。

ログから得られる情報からだけでは、不正行為の特定が解析できない場合や、大量ログへの対応について、現状で考えられる対策例を以下にあげる。

［ケース１：動的アドレス割り当てへの対応］

端末の識別方法としての端末名は、容易に書換えられるので IP アドレスを利用することが多い。しかし、組織で DHCP サーバ利用による動的な IP アドレス体系を利用している場合、特に IP アドレス数に余裕がない状態では、同じ端末でも異なる IP アドレスを割り当てられる可能性がある。IP アドレスがどの端末のものであるか判断するため、端末 MAC アドレスと紐づいている DHCP サーバのログを同時に保存し、識別に利用することが必要である。

［ケース２：データベースアクセスユーザの特定］

組織で利用している情報システムは、サーバクライアント型や Web 型とバックエンド DB を備えた形態が多くなってきている。サーバアプリケーションや Web アプリケーションは、クライアントを認証したユーザ名ではなく、DB アクセスするロールに対応した別ユーザ名に集約する場合がある。これは、クライアントユーザ数分の管理を DB のユーザ管理として行いたくないことが理由の場合が多い。また、アプリケーションが Java 言語で開発されている場合、異なる種類のアプリケーションから DB にアクセスしても DBMS から見えるアプリケーション名は唯一 Java になる。アプリケーションから実ユーザ名、実アプリケーション名などを DBMS に別途通知し、監査に有効なログを記録する製品（IBM:DB2 UDB）もあるが、プロセスを管理する OS レベルでの対応は実現できていない。

［ケース３：大容量ログへの対応］

監査ログは、通常の障害ログ、エラーログ、ポリシ違反ログと異なり全てのプロセスや統制の記録を行う必要がある。このためログの容量は膨大なものになり、ログの種別用途によって適切な保存期間を決める必要が生じる。また、監査証拠として証拠力の強い生ログの保管も大容量の保存装置を必要とする。このためログ容量を圧縮し、かつ圧縮したままでも検索できる機能を持つログ管理製品も提供されている[40]。

［ケース４：一定期間のログ保管］

監査ログを一定期間改ざんさせずに保存するためには、ログの変更・削除機能が存在しないことが望ましい。そのためには、1 度書き込んだデータを上書きや消去できないようにする WORM（Write Once Read Many）技術に対応したログ管理が必要となる。現在は、CD-R や DVD-R などの追記型記録媒体だけでなく、ハードディスクやテープの大容量ストレージにも WORM 機能を備えた製品が提供され､改ざん･削除防止を実現するログ管理ソリューションが提供されている。

業務プロセスに関わる内部統制として､アプリケーション統制と IT 全般統制が位置付けられている。アプリケーション統制では、アプリケーションで入力・処理・出力の不正やミスを防止するための、データのチェックや承認を行うための管理機能を定義している。IT 全般統制では、ソフトウェアの変更やハードウェアの運用に関わる管理機能を定義している。アプリケーション統制と IT 全般統制とに対応し、業務アプリケーションから出力されるアプリケーションログと IT プラットフォームから出力されるシステムログとを用いた 2 方向からの追跡を行うことで、監査証拠の証拠力を強めることができる。また、内部統制では、ユーザやそのアクセス権、ロールの作成、変更、削除、有効性のレビューは、承認プロセスを通じて実施することを要求している。これらのプロセスワークフローを備えた ID 管理製品として、SUN Java System Identity Manager、HP OpenView Select Identity、IBM Tivoli Identity Manager、Oracle Identity Manager などの製品がある。

またOSSのログ管理ツールとしては、ログを管理する目的よりは、ホスト型IDSのようにログを利用して監視する目的のものがある。例えばNagios[41]、OpenNMS[42]等である。さらに、これらのログ管理（監視）やネット性能監視、ジョブスケジューラツールを組合せ、そこにジョブ管理機能やパッチ適用などの一括制御機能を独自に追加した統合運用管理ツールHinemos[43]がある。Hinemosは、ログ取得もsyslog-ngなど既存のログだけでなく、Hinemosエージェントを利用してアプリケーションログの取得も可能であり、統合ログ管理としても利用できる。

2.5.2　フォレンジック

IT で発生するインシデント･レスポンス、IT に関わる訴訟、法的争いに対応する証拠収集、証拠保全を行う手法や技術をデジタル･フォレンジックという。デジタル･フォレンジック技術は、

監査ログで対応できない削除されたファイルの復元や、再初期化されてしまったハードディスクの復元などによって、証跡を解析・調査するもので、暗号化されている証跡の復元まで含まれる。その過程では、証拠保全を行い、証跡を壊さないように作業を行う。デジタル・フォレンジックと監査証跡を組合せることにより、アプリケーションの不具合、内部情報の漏洩、管理者の不正操作など組織に損害を与えるインシデントの発生手順を明確につかむことができ、より対策を強化することが可能となる。

例として図 2.17のように、監査対象組織が不利益な監査ログを削除してしまった場合でも、監査ログを保管するサーバ内のハードディスクにある情報をデジタル・フォレンジック技術により、証拠保全用および解析用の別ハードディスクに物理的に忠実に再現し、その解析用ハードディスクをさらにデジタル・フォレンジック技術によって解析を行えば、論理的に削除された監査ログが再現可能となる。再現できた場合、監査ログが暗号化されていれば、さらに復元を行う。

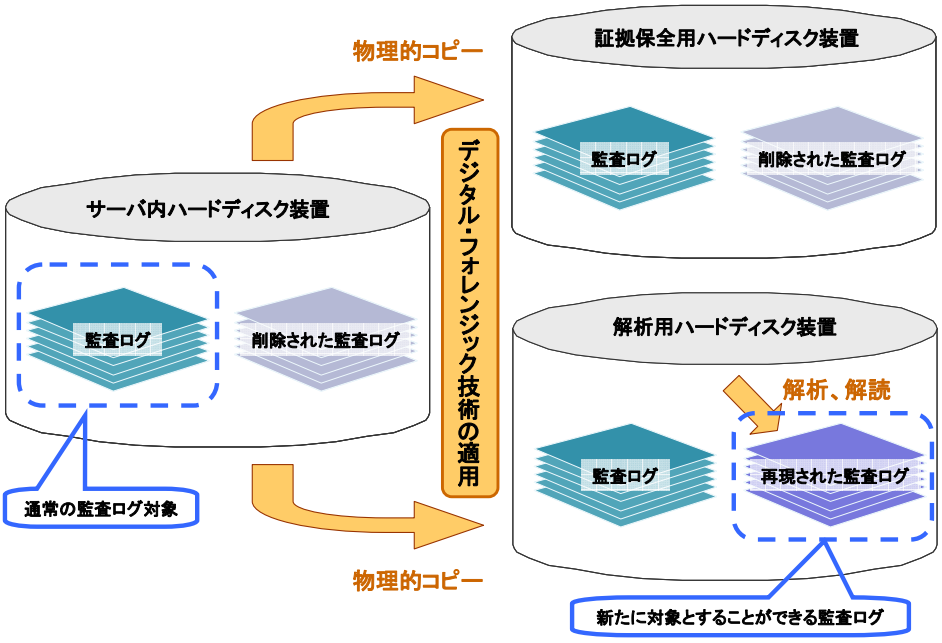

図 2.17　監査ログとデジタル・フォレンジックの例

2.5.3　大量のログの収集・分析

監査ログは内部統制の要請に基づき、システムを構成する様々な機器やコンポーネントから出力される、日々蓄積されるログを長期間記録する必要がある。しかし、日々膨張するログを長期間保管するためには、大容量のストレージを確保する必要があり、ストレージを確保するコストも莫大になる。また、システムの性能低下にもつながる。このため、必要なログ収集・管理を効率よく行なうことが必須の要件である。

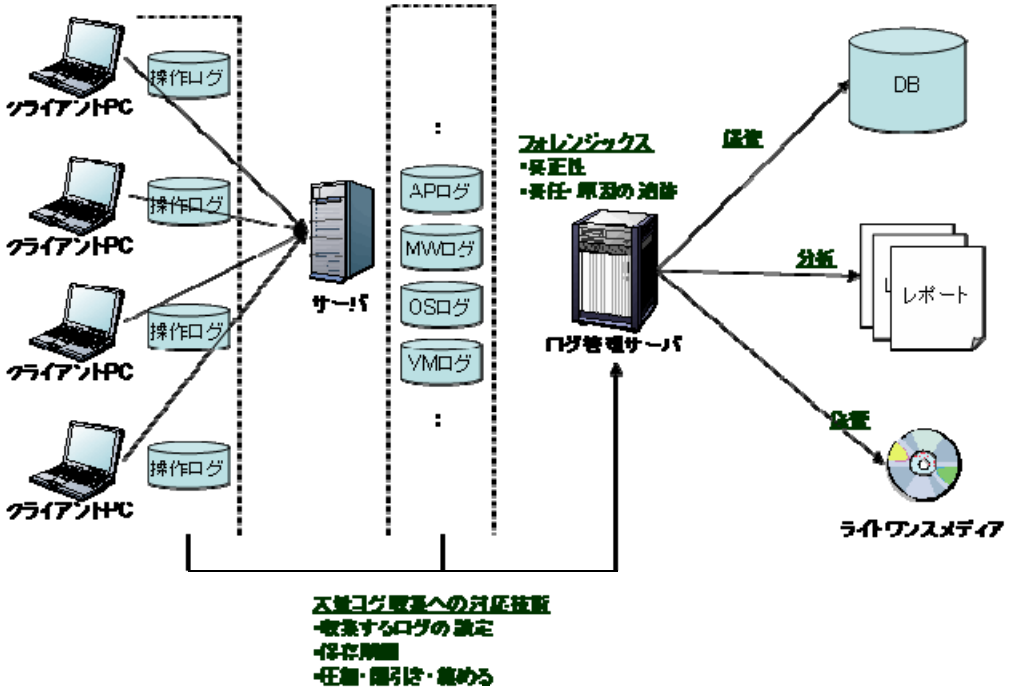

図 2.18　大量ログの収集・分析機能の概要

① 収集するログの選定

内部統制に基づき充分詳細なログを記録する必要がある。しかし、運用する中、システムを構成する様々な機器・コンポーネント（以降、リファレンスモニタ）からログは出力される。これらの全てを記録するとシステムの性能低下、大容量のストレージの確保が必要となり、現実的に運用することは難しい。まず、出力されるログの全てを洗い出し、必要なログを選定する必要がある。

この作業は、運用システムの構成に熟知していないと難しいため、多くのメーカは、コンサルティングサービスを用意している。

以下、収集するログの選定例を述べる。

表 2.5　収集するログの選定例

運用例	収集するログ
業務	認証の成功/失敗のログ
	ユーザーの追加/削除のログ
	コンテンツ(ファイル/電子メール)、DB へのアクセスのログ
	業務プロセス(操作、アプリケーション、ミドルウェアの

	ログ)
システム構築・保守	システム管理者の操作ログ
	プログラムファイルなどのアクセスログ
	システム（syslog）

② 保存期間

ログは、監査証跡として長期期間の保存が要求される。しかし、明確な保存期間の基準はなく、企業毎に規定していることが現状であり、各製品は、保存期間の設定および、期限が過ぎたログを破棄する機能を実装している。

以下、ログの保存期間の参考を述べる。

表 2.6　ログの保存期間

規定者	保存期間	備考
NIST	影響度高のシステムで３～１２ヶ月	頻繁に分析を行う事で、問題を発見を行うため、比較的短い時間と予想される
警察庁	３ヶ月以上を推奨	
	コンピュータ犯罪に関する刑法の時効　３～１０年	
HIPAA	５年	病院記録
	死後２年	医療記録
SEC 17a-4	３年	電子メール、財務記録
	口座廃止後６年	取引記録
Sarbanes-Oxley	監査後７年	監査記録

NIST: National Institute of Standards and Technology,

HIPAA:United States Health Insurance Portability and Accountability Act

SEC 17a-4:米国証券取引委員会（SEC）要求事項 17A－4

Sarbanes-Oxley:米国企業改革法

③ 圧縮

日々膨張するログを長期間保管するためには、大容量のストレージを確保する必要があり、ストレージを確保するコストも莫大になる。この問題を解決する方法として、圧縮技術が各ログ管理製品に実装されている。多くの製品は１／１０程度の圧縮性能を持つ。ここで採用されている圧縮方式では、圧縮したままの検索が可能となっており、伸張したデータを格納するストレージ領域を必要としない。一部１／４０程度の圧縮性能を持つものもあるが、圧

縮した状態での検索ができない。
ログのサイズをより小さくする技術として、重複しているログを間引く、複数のログを纏め１つのログに置き換える技術を実装している製品がある。
例えば、アプリケーションを操作しファイルの読み込みを行った場合、OS レベルでは、リードが何回か行われる。この時、一定期間内の同一のログが発生した場合、１つにまとめてしまう。また、ファイルの書込みの場合、リード、デリート、ライトの複数のアクセスが行われる。これらを纏めて１つのログ（上書き保存）とする。
保存するデータ量は、数百分の１と飛躍的に小さくなるものの、ログを収集した後、保存/レポーティングの際、「間引き・纏める」処理が行われている。「原本保管が望ましい」とされていることを考えると「不適切な変更が行われていないか」の証明が難しい。

④　ログの真正性
証拠保全のため、ログ管理製品では、ログの改ざん防止機能（暗号化）・電子書名の付与機能を提供している。また、ライトワンスのメディアへの出力により、改ざん防止を提供しているものもある。
ログ収集前のリファレンスモニタが出力したログに関しても真正性が必要であり、自コンポーネントの機能や、プラットフォームの機能を利用し、改ざん防止が必要である。

⑤　責任・原因の追跡
すでに多くのリファレンスモニタが出力するログには、４Ｗ１Ｈが含まれており、ログ管理製品はそれを保存している。責任や原因を追跡するため、意味付け（定義ファイル）や、異なるログ形式の横断的な検索機能を実現している。
また、多くのログ管理製品は、圧縮したままの検索を可能としており、数百万件／秒程度の検索性能を持つ。
なお、ログには、Who（誰が）、When（いつ）、Where（どこで）、What（何を）、How（どうした）を含む情報が出力される。また、監査ログでは、異常や拒否だけではなく、成功や許可も出力される。

⑥　収集
ログが出力される場所はサーバやネットワーク機器など分散されており、データ形式も一律ではない。多くのログ管理製品は、様々なデータ形式（システムログや、イベントログ、ネットワーク機器固有ログ）とプロトコル（ftp/syslog/エージェントインストール）をサポートしている。形式が不明なものに関しては、変換ツールを準備している。

⑦　分析
分析機能の主な機能は、目的別のテンプレートが準備されている。以下にその例を述べる。

- ログインやファイル I/O の失敗回数・状況レポート
- 夜間の利用回数・状況レポート
- 機密ファイルへのアクセス失敗回数・状況レポート

Linux に含まれる logrotate というプログラムは、指定されたタイミングでログファイルのバックアップを取り、新しいログファイルを作成する。これにより、ログファイルの容量が肥大化することを防ぐ。

統合的なログ管理の代表的な製品に、三菱インフォメーションテクノロジーの統合ログ収集・分析システム LogCatcher シリーズやインフォサイエンスの統合ログ管理システム Logstorage がある。これらの製品は、機器の種類、アプリケーション、フォーマットを問わず、サーバやネットワーク機器が出力するログを統合的に管理する。ログの検索による原因調査、ログの集計による監査、リアルタイムのログ検知によるログ監視、自動定期レポートによるログ監査、自動定期ログアーカイブといった機能を実現している。

2.5.4 監査ログに対する技術要件

監査ログに対する技術要件を表 2.7にまとめた。

監査ログに対する上記の要件をふまえ、監査ログの統合管理にむけた技術課題を整理する。

① 監査ログに記録する情報の形式は、さまざまな形式が存在するため、複数の監査ログに対して一括したフィルタリング、検索、統計処理が行えない。この対策として、ログ管理ツールで正規化したツール指定のフォーマットに変換し、ログを蓄積する技術がある。この変換フォーマットは、異常検知のパターンマッチングや検索などテキスト処理に適した形式が多く、XML 化されていることは少ない。製品から出力される監査ログ、および個々の監査ログをまとめた統合監査ログの相互運用性確保が必要になる。

② 個々の製品で監査ログを記録するポリシの管理は、統合ログ管理ツールではなく、ポリシ管理ツールによっている。現状の監査ログポリシは、ポリシ管理ツールによって確認できるが、過去の監査ログが採取された時点でのポリシは記録に残っていない。ポリシ管理ツールと統合ログ管理ツールの連携が必要である。これは①とともに、複数のログにまたがって一連の処理や操作の追跡ができない原因となっている。

③ 通常のポリシ管理ツールは、複数製品にまたがる監査ログポリシを一括して設定する場合に、個々の製品の設定機能と調整を行う必要があるため、管理対応製品を限定している。このことは、組織内全ての監査ログを一元管理することに対しての障害になっている。製品側および監査ログポリシ管理側に共通した統合監査ログポリシ記述とその処理定義が必要である。

④ 監査人から必要な監査ログの提出を求められた場合、すみやかに提出しなければならない。検索性能を上げるために、監査ログを書換え不可な記録装置でなくハードディスク装置に保存して可用性を向上させると、逆に改ざんが容易になってしまう。ある時刻における存在と

非改ざんの証明ができるタイムスタンプ技術が必要になるが、タイムスタンプ技術だけでは、タイムスタンプを押す単位での監査ログの証明が可能としても、押す単位間の関係は時刻による前後関係でしか証明できない。存在する監査ログの間に監査ログが抜けていないことを証明する必要がある。

表 2.7　監査ログに対する技術要件

監査ログ内容	1-1)ユニークなユーザおよびリソースの識別ができること。
	1-2)システムが正しく動作していることが追跡できる情報が記録されていること。
	1-3)不正行為が検知できる情報が記録されていること。
	1-4)イベントが発生したことを記した時刻が正しいこと。
監査ログ採取	2-1)取得されたログは、一元・集中管理されること。
	2-2)大量のログに対する対策（ログサイズ、転送トラヒック量、ロストなどの対策）を行うこと。
	2-3)採取するイベントの種類や内容を選別できること。
監査ログ利用	3-1)リアルタイム監視にもログが利用可能なようにすること。
	3-2)定期的な内容チェックができるように、解析、統計ができること。
	3-3)監査報告用のレポートができること。
	3-4)非定型検索を適切な時間内に行えること。
監査ログ保存	4-1)定期的なバックアップを行い、障害時に備えること。
	4-2)監査に十分な期間だけでなく、監査対応期間終了後もインシデントが発見される可能性があるために、十分な期間、保管管理できること。
	4-3)電子署名やタイムスタンプを使用している場合、長期保存対策をすること。
	4-4)所定の期間、監査ログ原本を削除できないこと
監査ログのセキュリティ	5-1)システム及びアプリケーションで採取したログは、削除、改ざんを受けることなく正しくログデータベースに保存できること。
	5-2)保存されたデータベースは、適切なアクセス制限を行うこと。
	5-3)保存されたデータベースは、アクセス履歴を記録すること。

	5-4)保存されたデータベースは、改ざんや破壊に対して 対策を行うこと。
	5-5)保存されたデータベースと同じ内容のデータを出力でき、 同じ内容であること確認できる改竄検知コードをデータと関連付けて保存できること。

2.5.5 まとめ

- 内部統制や各種ガイドラインの要求により、大量な監査ログを効率的に収集、記録、検索する必要性が増大している。監査ログの統合管理には、多種類のログへの対応、および監査ポリシ管理やログデータベースを含めた管理システム相互運用などの課題を解決し、高信頼性の監査ログを有効的に利用できるプラットフォームを構築することが必要である。
- ログ管理の製品には、三菱インフォメーションテクノロジーの LogCatcher シリーズやインフォサイエンスの Logstorage がある。これらの製品は、機器の種類、アプリケーション、フォーマットを問わず、サーバやネットワーク機器が出力するログを統合的に管理する。ログの検索による原因調査、ログの集計による監査、リアルタイムのログ検知によるログ監視、自動定期レポートによるログ監査、自動定期ログアーカイブといった機能を実現している。

2.6 統合管理

内部統制の実現の為に、ID 管理、アクセス権管理を統合的に扱う製品がでてきている。ID 管理とアクセス権管理を統合的に扱うことにより、組織のセキュリティポリシを遵守の実現が可能となると共に、アクセス権の設定状況の把握が可能となる。

Beta SysytemsのSAM Jupitorなどの、ID管理とアクセス権管理を統合管理できる製品が出始めてきている。アクセス権管理においては、人毎の管理を行なうことによる作業量の増大をさける為に、ロールベースでのアクセス権管理であるRBAC(Role Based Access Control)にて管理を行なう製品が出てきている。ID管理の部分においては、2.4.3で述べたSun Java Identity Managerのように、SoD (Segregation of Duties:職務分掌)のチェックを行なえる製品が出てきている。

2.6.1 SAM Jupiter

SAM Jupiter[58]は、次のコンポーネントから構成されている。

① Provisioning Server
ユーザのライフサイクル管理の主要コンポーネント。人事のリソースや企業のディレクトリから、関係するすべてのシステムにおけるユーザアカウントとアクセス権の作成、削除するアクションに変換する。

② Security Administration Client
ユーザ、ロール、リソースについての情報の管理機能。エンタープライズレベルの管理とターゲットシステムレベルの管理が統合される。

③ Delegated Administration Facility
組織構造などを考慮した委譲機能を提供する。

④ Policy Enforcement Facility
定義されたセキュリティポリシに従った設定にて動作しているかを監視し、ポリシ違反を検出/修正する。

⑤ Reporting and Auditing
ユーザに現在割り当てられているアクセス権のスナップショット、セキュリティポリシにおける監査リスト、アラートのリストのスナップショット。HTML、PDF、Excel のファイルで出力が可能。

⑥ Help Desk
パスワードを忘れたエンドユーザに対するパスワードリセットなどの機能を提供する。

SAM Workflow は、SAM Provisioning Server とシームレスに連携するワークフローエンジン。

SAM Connectors は標準のコネクタで、Windows2000/2003、NT、IBM AIX、IBM AIX NIS、OS/400、HP-UX、HP-UX NIS、Linux (via LDAP)、Sun Solaris、Sun Solaris NIS, NIS+、Novell NetWare、MS Exchange、Lotus Domino、Novell GroupWise、RACF、ACF2、TopSecret、TSO、Tivoli Access Manager (TAM)、LDAP、Oracle、DB２、SAP R/3 をサポートしている。

SAM Role Modeler (2007 年 6 月リリース。元は、SAM Role Miner という製品)は、ビジネスプロセス分析からロールをマイニングするツール。ここで作成したロールを SAM Jupiter へ渡す。

SAM Log Audit Facility は、SAM Jupiter のトランザクションログから、管理者の実行した時間帯や管理者名、ビジネスオブジェクト名、インサート、変更、削除などの動作、影響するユーザなどの情報などのレポートを作る。

SAM Jupiter は、管理モデルとしては、RBAC モデルを利用している。

また、管理操作が行える範囲を示すスコープ(scope)と呼ばれる概念を導入している。管理者は、スコープ内の特定の部門のみを管理することができる。スコープを活用することにより、階層化管理を実現することが可能となっている。例えば、図 2.19のような組織構造を考える。図の中の色を付けた範囲が、ある管理者のスコープを表す。

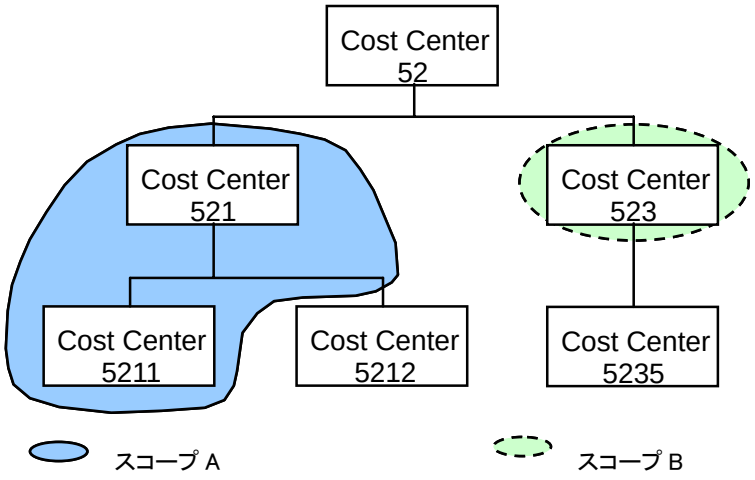

図 2.19　組織構造の例

表 2.8に、図 2.19の組織構造に対応したSAM Jupiterのスコープを示す。スコープは、頂点で示された組織を基点として、Cost Center523 のような単一のノード（Node）のみであるか、Cost Center521 のような木構造（Tree）であるかを表す。Cost Center521 のような木構造の場合、頂点で示された組織Cost Center521 を基点とした階層的な組織構造を表す。排除（Exclude）がチェックされているCost Center5212 のような組織構造は、スコープの範囲外とする。

表 2.8　図 2.19の組織構造に対応したスコープ

スコープ	Node	Tree	Exclude	頂点
A		○		Cost Center 521
	○		○	Cost Center 5212
B	○			Cost Center 523

SAM Jupiter では、関連製品として Role Modeler があり、ロール管理の支援を行なえるようになっている。

Role Modeler (旧 Role Miner)では、ゼロからロールを構築するのではなく、既存のシステムにあるセキュリティ情報を使って、ボトムアップでロールを構築することを支援するツールである。を利用している。これは、既存のセキュリティ情報に対して、データマイニングの技術を使って、クラスタリングや相関分析を行なうことにより、ロールを定義するための属性の候補を決めていく。但し、完全に自動化が可能なわけではなく、途中段階での関与や、結果のチェックは必要となっている。

基本的なアルゴリズムのフローは以下のようになる

1. インタラクティブな改良プロセスにおける階層的なクラスタリング

2. クラスタの最大数を与えられるとき、partioning clustering を実行する。
3. クラスタの質を評価するために Condorcet の基準を使う
4. 離散の変数と同等の量を扱う。離散変数として、数値データを扱う。
5. 特定の属性の類似と相違によって作用する属性の重み付けを実行する

【適用例】

ケース A：生産性に寄与した事例企業

この企業は、ドイツの保険会社で 14 万人の従業員が世界中にいる。１万８千人のユーザ ID に対して、930 の SAM モデルが存在する。これらのモデルは、セキュリティ部門で長期間に渡って開発され、手動で管理されていたため重荷となっていた。SAM Role Miner は、SAM データベースに蓄積されたデータを分析するのに使われた。何時間かかけて、930 すべてのロールが決定することが出来た。SAM Role Miner を使って、ロールエンジニアリングに要する時間が、月レベルから時間レベルに改善された。

ケース B：ビジネスに利益をもたらした事例企業

この企業には約 400 のロールがある。手動によるロール生成に１ロールあたり５千ドルかかり、１ロールあたりの保守コストは年間 2,500 ドルである。ロールを編成し直すまでの期間は、３年と見積もっている。ロール数の増加率を年５％と仮定すると、ロールエンジニアリングのコストは 200 万ドル、新しいロールの生成とロールの保守にかかる年間コストは 433,000 ドルで予測されている。ロールマイニングを使うと、最初のロール生成の約 60%をセーブし、保守の約 50%を減らせると想定できる。

【適用による課題】

クラスタを発見するところでまだ大きな手作業の介在が必要。ロール発見プロセスの 80 から 90％の自動化に対しては、このアプローチでは現実的ではない。従って、ロールマイニングプロセスとトップダウンビジネスプロセス分析アプローチの融合を考えていく必要がある。

2.6.2 まとめ

SAM Jupiter の対象範囲は、コネクタの提供により広範囲を対象としている(各種 OS、各種アクセス制御製品、DB、アプリ、ワークフロー)。ただし、VM は対象となっていない。管理手段は、RBAC による管理、階層化管理である。制御項目は、明確な情報はないが、対象が広いため、様々な制御が可能と推測できる。制御手段に関しては、SAM Jupiter として制御を強制する機能を持つ訳ではない。設定を監視し、問題がある場合には、対応を促すものである。実際の制御に関しては、各 OS の機能、アクセス制御製品と連携して行なう。

3. まとめ

サーバシステム統合管理に関わる要素技術について述べた。内部統制の観点から。所定のセキュリティポリシに基づいてアクセス制御を正しく行うことと、それを容易に確かめることが、サーバシステム統合管理の要件として注目されている。セキュリティポリシのサブジェクトに関する ID 管理、オブジェクトに関するリソース管理、サブジェクト、オブジェクトにアクションを含めたアクセス制御に関わるポリシ適用、アクセス制御が正しく行われたことを確かめるためのログ管理（監査）についての技術動向を説明した。技術動向の要点を以下にまとめる。

- ID 管理に関して、ID プロビジョニングに用いる SPML、SPML に関連する DSML、SAML、ID-FF といった規格の標準化が進められている。さまざまなベンダが提供するさまざまな種類の管理ターゲットを一元的に管理するための相互運用性の実現にむけて、開発が進んでいる。ID 管理単独では、技術が比較的成熟してきており、新しい動向は少ない。
- リソース管理に関して、DMTF で CIM と WBEM が開発されている。CIM の仮想マシンモニタへの拡張や WBEM にもとづく情報システムの運用管理全般の標準化などが推進されており、Xen-CIM や OpenPegasus などのオープンソースの実装の開発、XML をベースとした検索技術の開発などが進んでいる。WS-Management と関連仕様群は、WinRM（Windows Remote Management)、Wiseman、Openwsman といったさまざまな商用製品、オープンソースソフトウェアで利用可能となっている。ITILv3 に従う構成管理の技術として CMDBf（CMDB Federation）がある。IBM Tivoli CCMDB、HP Universal CMDB、CA Unicenter CMDB といった多くの商用製品が、ITILv3 の要件を満たす CMDB を提供している。
- ポリシ適用に関して、XACML の RBAC ポリシへの拡張、VM や OS において RBAC ポリシを実施するためのリファレンスモニタの開発が進んでいる。ロール階層化とロール割り当ての制約を付加した RBAC は、職務分掌を実現するためのモデルとして注目されている。OS には、リソース使用量の制限を設定することで、ユーザやプロセスが占有するリソース使用量を制御する機能を持つものがある。Linux では、setrlimit、CKRM、CABI などが使用できる。Windows では、BES、Win32API、System Service などが使用できる。
- ログ管理に関して、大量で多くの種類の監査ログの効率的に統合管理、ポリシやログデータベースを含めた管理システム間の相互運用、信頼性の高い監査ログを有効的に利用できるプラットフォームの構築といった課題が注目されており、開発が進められている。ログ管理の製品には、三菱インフォメーションテクノロジーの LogCatcher シリーズやインフォサイエンスの Logstorage がある。これらの製品は、機器の種類、アプリケーション、フォーマットを問わず、サーバやネットワーク機器が出力するログを統合的に管理する。
- 統合管理が可能な製品には、SAM Jupiter や Sun Java Indentity Manager/System Access Manager がある。統合管理が可能な製品が登場し始めた段階にあり、実際の制御に関しては、各 OS の機能やその他のアクセス制御製品と連携して行なう。特に、VM や NW 機器との統合管理は、今後の課題である。

参考文献

[1] J. Lewis, “Enterprise identity management: It's about the business,” Burton Group.

[2] http://sdc.sun.co.jp/javasystem/techtopics/identity/200702.html.

[3] OASIS Service Provisioning Markup Language Version 2.0.

[4] http://www.oracle.co.jp/products/middleware/identity-management/identity-management.html.

[5] http://www.projectliberty.org/jp/resources/LAP-ID-FF-archtecture-overview-v1.2-JP.pdf.

[6] http://www.oasis-open.org/committees/download.php/17708/pstc-spml-2.0-os.zip.

[7] http://www.oasis-open.org/committees/dsml/docs/DSMLv2.doc.

[8] http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip.

[9] http://www.projectliberty.org/jp/resources/LAP-ID-FF-archtecture-overview-v1.2-JP.pdf.

[10] http://www.dmtf.org/standards/cim/.

[11] http://www.wbemsolutions.com/tutorials/CIM/.

[12] http://www.xensource.com/files/summit_3/XenCIM.pdf.

[13] http://www.wbemsolutions.com/tutorials/DMTF/wbem.html.

[14] http://safari.informit.com/9780132349710/ch11lev1sec6.

[15] http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpdnwmi/htm/dmtf.asp.

[16] W3C Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/xml.

[17] W3C XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath.

[18] W3C XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt.

[19] W3C XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery.

[20] DMTF DSP0202, CIM Query Language Specification Version 1.0.0, http://www.dmtf.org/standards/published_documents/DSP0202_1.0.0.pdf.

[21] The Global Alliance, http://www.globus.org.

[22] P. Loscocco *et al.*, “Integrating flexible support for security policies into the Linux operating system,” in Proceeding of FREENIX (2001).

[23] L. Gong *et al.*, “Implementing protection domains in the Java development kit 1.2,” in Proceedings of Internet Society Symposium on Network and Distributed System Security (1998).

[24] L. Badger *et al.*, “Practical domain and type enforcement for UNIX,” in Proceedings ofthe IEEE Symposium on Security and Privacy (1995).

[25] D. E. Denning, “A lattice model of secure information flow,” Communication of ACM, vol. 19, pp. 236–243 (1976).

[26] R. S. Sandhu *et al.*, “Role-based access control models,” IEEE Computer, vol. 29, pp. 38–47 (1996).

[27] http://www.xrml.org/.

[28] S. Oh, *et al.*, “Task-role based access control (T-RBAC): An improved access control model for

enterprise environment, in Prococeedings of the 11th International Conference on Database and Expert Systems Applications (2000).

[29] 森田陽一郎ほか、安全なアドホック情報共有のための動的ACL設定システム、第３回情報科学技術フォーラム（FIT2004）.

[30] J. Park *et al.*, “The UCONABC usage control model,” ACM TISSEC, vol. 7, pp. 128-174 (2004).

[31] X. Zhang *et al.*, “A usage-based authorization framework for collaborative computing systems,” In Proceedings of ACM Symposium on Access Control Models and Technologies, (2006).

[32] http://seedit.sourceforge.net/doc/2.0/spdl_spec/

[33] N. Damianou *et al.*, “The Ponder policy specification language,” Lecture Notes in Computer Science, vol. 1995 (2001).

[34] J. DeTreville, “Binder, a logic-based security language,” in Proceedings of IEEE Symposium on Security and Privacy (2002).

[35] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[36] http://www-03.ibm.com/servers/eserver/zseries/zos/racf/.

[37] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[38] http://jp.sun.com/products/software/identity/identity_mgr/.

[39] http://jp.sun.com/products/software/identity/access_mgr/.

[40] http://www.sensage.com/.

[41] http://www.nagios.org.

[42] http://www.opennms.org.

[43] http://sourceforge.jp/projects/hinemos/.

[44] D. F. Ferraiolo *et al.*, “Proposed NIST standard for role-based access control,” ACM Transaction on Institute and System Security, vol.4, pp.224-274 (2001).

[45] R. S. Sandhu. *et al.*, “The ARBAC97 model for role-based administration of roles”, ACM Transactions on Information and System Security, vol. 2, pp. 105-135 (1999).

[46] Joon S. Park, *et al.*, “A Composite RBAC approach for large, complex organizations,” In Proceedings of ACM SACMAT (2004).

[47] A. A. El Kakam, *et al.*, “Organization based access control,” in Prodeedings of IEEE International Workshop on Policies for Distributed System And Network (2003).

[48] http://seedit.sourceforge.net/doc/2.0/spdl_spec/.

[49] http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[50] http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf.

[51] R. Sailer *et al.*, “Building a MAC-based security architecture for the Xen opensource hypervisor,” in Proceeding of ACSAC (2005).

[52] http://seedit.sourceforge.net/.

[53] http://jp.sun.com/products/software/identity/identity_mgr/.

[54] http://jp.sun.com/products/software/identity/access_mgr/.

[55] http://www.cisco.com/japanese/warp/public/3/jp/product/hs/ifmodule/csm/prodlit/pdf/cace_pb.pdf.

[56] NPO デジタル・フォレンジック研究会, http://www.digitalforensic.jp/.

[57] 情報セキュリティ管理基準 Version 1.0、経済産業省.

[58] Beta Systems Software AG, SAM Jupiter, http://ww2.betasystems.com/en/products_new/security/.