低コストで実現するディスクロージャー・マネジメント

― 個人情報保護のための新たなアクセス制御モデルの実現 ―

河岡 忠広　住田 敦

Disclosure Management Realized at Low Cost.

– The New Access Control Model for the Protection of Personal Information –

Tadahiro Kawaoka　Atsushi Sumida

個人情報保護におけるディスクロージャー・マネジメント（情報開示管理）に要求されるアクセス制御要件では，プライバシーポリシーに基づくアクセス制御が必要となる．これを実現するには，“ ユーザカテゴリ ”，“ データカテゴリ ”，“ 目的 ”，“ 条件 ”，という4つの要素に基づくアクセス制御が求められる．これらの要素に基づくアクセス判断に必要な情報は，DB内の利用者や情報主体の属性情報や，認証サーバが利用するユーザレポジトリ，および個人情報アクセス時に利用者から渡されるデータ内などに存在している．これらの情報を取得できる一番現実的なポイントはアプリケーションである．従って，アクセス制御はアプリケーションで実施することが最も現実的と言える．しかしながら，アプリケーションでアクセス制御を実施することの問題は，アクセス制御に必要な機能であるPDP（Policy Decision Point：ポリシー決定点）とPEP（Policy Enforcement Point：ポリシー実施点）をアプリケーション内で実装することに伴うアプリケーション開発のコストとポリシーの一貫性喪失のリスクである．この問題を解決するにはアクセス制御機能をなるべくインフラ側に任せることが有効である．本論文では，新たな解決策として，Tivoli® Access Manager for e-business（TAMeb)のアクセス制御機能を活用することで，低コストでこれを実現する手法を提示する．

Disclosure management to protect personal information requires access control based on a privacy policy. Access control should be managed according to four factors: "user category", "data category", "purpose", and "condition". The information required for making access control decisions depending on these factors is found in the attributes of users, those of information items stored in the user repositories used by the authentication server and in the data passed from the users when they access personal information. The most realistic way to access these pieces of information is to use application programs. Therefore, it can be considered the most realistic for access control to be performed by application programs. However, we have a problem with access control performed by application programs because there are risks of higher development costs and consistency loss if Policy Decision Point (PDP) and Policy Enforcement Point (PEP) are implemented in application programs. An effective solution to this is to leave access control as much as possible to the infrastructure. This paper proposes as a new solution a technique to realize this at low cost by utilizing the access control function of Tivoli® Access Manager for e-business (TAMeb).

Key Words & Phrases：ディスクロージャー・マネジメント，個人情報，プライバシー，アクセス制御，
Tivoli Access Manager for e-business
Disclosure management, personal information, privacy, access control,
Tivoli Access Manager for e-business (TAMeb)

提出日：2005年08月31日　再提出日：2006年3月24日

1. はじめに

個人情報漏えい事件が相次ぐ中, 個人情報保護に対する意識がこれまでになく高まっている. また, 2005年4月に, 個人情報保護法案が施行され, 企業は自身が保有する個人情報を保護するための, 高度な体制を確立し維持していく必要にせまられている. 対応策としては, ITシステムでの対応, 運用・プロセスでの対応など様々な方策が考えられるが, その解決策の一つとしてディスクロージャー・マネジメントがある. これは, 利用者に, 業務に必要のない個人情報は開示しないという方針で, 情報漏えいを最小限に食い止めることを目的とするソリューションである. ディスクロージャー・マネジメントの具体的な対応策は多岐に渡る[1]が, 本論文では, 個人情報を参照する際のアクセス制御にフォーカスを当てる. このアクセス制御の実現方式としては, 通常, 業務アプリケーションなどで実装する形態が一般的である. しかし, このアプローチでは, 既存のシステムにこの機能を付加する際におけるアプリケーションの改修に伴うコストや, ポリシーの一貫性を維持することが課題となる. 本論文ではこれらの課題を解決する手段の一つとして, TAMeb(Tivoli® Access Manager for e-business)[2] [3]を利用した, フロントエンドで許可判断を行う新しい解決策を提示する. なお, TAMebとはWebシステムのフロントエンドに設置され, 認証認可をつかさどるリバースプロキシサーバである.

以下, 2章でアクセス制御の要件と, アクセス制御に必要な機能をまとめる. 3章では, 各種のアクセス制御モデルを紹介するとともに, 提案モデルを提示してそれらの特徴を整理する. 4章で提案モデルの詳細を述べ, 5章でTAMebによる実現方法の詳細を述べる. 最後に6章で, 本論文のまとめと今後の課題を述べる.

2. アクセス制御要件

2.1 アクセス制御に求められる要素

個人情報を保護するためのアクセス制御では, プライバシーポリシーに基づく制御が求められる. プライバシーポリシーとは, 個人情報を適切に取り扱うための指針であり, アクセス制御の側面としては,「誰が」「誰の」「どの」情報に「どのような目的」でアクセスできるかという視点から, ポリシーに照らし合わせて判定する機能が求められる[4]. 例えば,“ 配送部門は, 首都圏在住のお客様の住所, 電話番号を, 配送の目的で参照できる. ”といったポリシーや“ マーケティング部門は, お客様が同意する場合(Opt-in)には, お客様の年齢と購買履歴情報を購買傾向分析の目的で参照できる. ”といったポリシーに基づく制御が必要となる. これを実現するには, **図1**のようなデータレベルでのきめ細かい単位での制御が求められる.

図1. アクセス制御の例

アクセス制御要件の洗い出しには, EPAL(Enterprise Privacy Authorization Language)[5] [6]のルール構造が参考になる. EPALとはIBMが2003年7月に発表したものであり, 企業内でプライバシー管理の実施を目的としたきめ細かいポリシーを表現するための言語である. EPALのルールとして, 以下の要素が定義されている[4].

(1)ルール(Ruling):“ 許可する ”あるいは“ 禁止する ”といったルールに対する取り扱い方針を指定する.

(2)ユーザカテゴリ(User-Category):例えば, 組織の構成員や部署など, 個人情報にアクセスするユーザやグループを指定する.

(3)アクション(Action):個人情報がどのように取り扱われるのかを指定する. 例えば, 個人情報がDISCLOSE(開示)されるのか, STORE(保管)されるのか, あるいはDELETE(削除)されるのかなどを定義する. EPALでは, 個人情報のライフサイクル全般に渡ったルールの策定が可能である. 本論文においては開示管理の観点からDISCLOSEのみを対象とする. 従って, 当“ アクション ”はアクセス制御要件の対象外とする.

(4)データカテゴリ(Data-Category):データカテゴリは, 個人情報の持ち主である情報主体の保有する個人情報識別である. 例えば, 氏名, 住所, 電話番号, Eメールアドレスなどがそれに相当する.

(5)目的(Purpose):“ 目的 ”は, 個人情報が取り扱われる際の利用目的を表現する. 例えば, マーケティング目的なのか, 配送目的なのか, 決済目的なのかということを指定する.

(6)条件(Condition):“ 条件 ”は, コンテキスト情報をルール式で判断するために使用する. 例えば, 参照するユーザが, 東京在住であるかどうか, 情報主体が13歳以上であるかどうか, あるいは, 情報主体が利用目的に対して同意しているかどうか,

などのような条件式を記述することができる.

(7)義務(Obligation):個人情報の取り扱いについての義務を記述する. 例えば,“ 個人情報取得後30日経過したらその個人情報を削除する. ”といった内容や,“ 個人情報を取得したらメールで通知する. ”といった内容を記述する. アクセス制御では, この“ 義務 ”についての内容は直接関連しないため, アクセス制御要件の対象外とする.

以上, EPALの観点からまとめると, 個人情報保護のためのアクセス制御の要素としては,“ ユーザカテゴリ ”,“ データカテゴリ ”,“ 目的 ”,“ 条件 ”, の4つの要素に対して“ 許可 ”もしくは“ 禁止 ”の“ ルール ”を付与する形となる. これらの要素で表現したルールの一例を**表1**に示す.

表1. EPALで表現したプライバシーポリシー

No.	EPAL Rule	ルール1	ルール2
1	Rulling	許可する	許可する
2	User Category	配送部門	マーケティング部門
3	Action	参照する	参照する
4	Data Category	住所, 電話番号	年齢, 購買履歴
5	Purpose	配送	購買傾向分析
6	Condition	首都圏在住のお客様	Opt-in
7	Obligation	N/A	N/A

2.2 各要素に対する情報の存在場所

次に, この個人情報保護のためのアクセス判断に必要な情報であるADI(Access Control Decision Information)がどこに存在するかを考えてみる. ADIとはポリシーに照らしてアクセス判断する際に, 必要となるパラメータ情報のことである. このパラメータ情報としては, 前述のEPALのルールの7要素の内の“ ユーザカテゴリ ”,“ データカテゴリ ”,“ 目的 ”,“ 条件 ”の4要素が該当する.

(1)ユーザカテゴリ:アクセスするユーザの識別情報は, 認証情報の中にある. アクセス制御をするには, アクセスするユーザが正しく認証されていることが前提である. 認証時に認証情報が生成され, その中のユーザ識別情報がアクセス制御システムへ渡される必要がある.

(2)データカテゴリ:どの個人情報にアクセスするのかという識別情報は, 例えばユーザインターフェースである業務画面から指定する場合, クライアントから渡されることになる. また, アプリケーション側でどの個人情報にアクセスするかといった情報を持つ場合もある. この場合, 個人情報識別がクライアントから渡されることはない.

(3)目的:目的情報は, ユーザからパラメータとして渡される場合もあれば, ユーザがアクセスするアプリケーションそのものが利用目的となる場合もある. このときWebシステムではURLが目的情報となる.

(4)条件:条件のパタンとしては, 大きく3種類存在すると考えられる. まず一つめは, 利用者の属性情報を条件とする場合である. この場合, 属性情報は, ユーザレポジトリの中に存在する場合がある. あるいは, 利用者DBのようなバックエンドDBに格納されている場合もある. 2つめは, 情報主体の属性情報を条件とする場合である. その場合には, 顧客DBのようなデータベースに格納されていることが一般的である. 3つめは, 利用者の属性情報と情報主体の属性情報の両方を必要とする場合である. 各データの置き場所は前述したとおりである. DBに存在する場合には, そのDBからSQLでアクセス判断に必要な情報を取得する必要がある.

2.3 アクセス制御に必要な機能

アクセス制御モデルを検討する際には, 以下の機能をどこに配置すべきかを決めていく必要がある.

(1) PDP(Policy Decision Point):PDPとは、許可判断機能であり, 後述のPEPからアクセス判断に必要な情報を受け取り, アクセス制御ポリシーに従って, アクセス要求が正しい権限を持つものかどうかを判断し, 許可/不許可の決定(アクセス判断)を下すコンポーネントである. PDPは, 一般的にPEPから渡されるADIから, 許可判断を下さなくてはならない.

(2) PEP(Policy Enforcement Point):PEPとは、アクセス制御実施機能である. アクセス判断に必要な情報であるADIをPDPに渡し, PDPからのアクセス判断結果を受け取り, その結果に基づいてアクセス制御を実施するコンポーネントである. PEPは, アクセス制御に必要な情報であるEPALルール要素, つまり“ ユーザカテゴリ ”,“ データカテゴリ ”,“ 目的 ”,“ 条件 ”に相当する値を取得し, それらをPDPへ渡さなければならない.

3. アクセス制御モデル

アクセス制御モデルとして, SQLの結果をフィルタリングする形態と, 実行するSQLを使い分ける形態があり, それを実現するための複数の制御モデルが存在する. 以下に, 提案する新しい制御モデルも含めて各モデルの概要と特徴を述べる.

3.1 SQLの結果のフィルタリング

アプリケーションがDBにアクセスしその結果を取

得した後で, その結果情報をEPALルールによりフィルタリングする方法である. このモデルの実現方法としては, PDP, PEPともにアプリケーションで実装する方法が考えられる.（**図2**）この方法は, 認証機能, クライアントから渡される情報, アプリケーション, DBなどに存在するADIを入手することが容易である. ただし, アプリケーション毎にPDPとPEPを実装しなくてはならず, 許可判断に必要なEPALポリシーもアプリケーション毎に管理しなくてはならないため, 実装および運用のコストが大きい.

別の方法としては, アプリケーションの負担を軽減するためにPDPをアプリケーションから分離する形態である.（**図3**）このモデルを採用することにより, 複数のアプリケーションがPDPへ許可判断を仰ぐことが可能となる. それにより, ポリシーの管理を一元化することができる.

また, **図4**のように, PDPとPEPの両方をアプリケーションから切り離す形態が考案されている. 例えばIBMの東京基礎研究所ではMonitoring Tool for JDBCというソリューション[7][8][9]を開発している. このソリューションでは, PEPを行うJDBC（Java™ DataBase Connectivity）クラスを提供することで, アプリケーションからPEPを切り離すことに成功している[2]. これにより, アプリケーションのコードの改修が不要もしくは必要な場合であっても最小限で済むことが大きな特徴であり, 既存システムに組み込む場合には, 改修に伴う影響を極小化できるソリューションである.

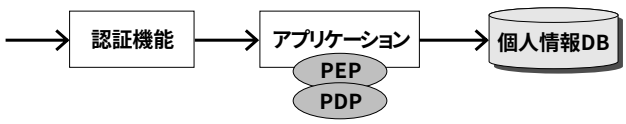

図2. モデル1-1 アプリケーションでPDP, PEPを実施するモデル

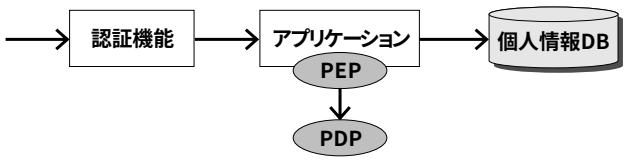

図3. モデル1-2 PDPをアプリケーションから分離するモデル

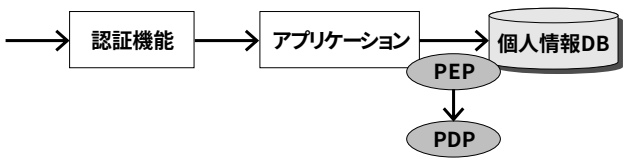

図4. モデル1-3 PDPとPEPをアプリケーションから分離するモデル[5]

3.2 実行SQLの使い分け

アプリケーション内でDBにアクセスするためのSQLを複数用意しておき, EPALのルールに基づいて, そのSQLを使い分けるモデルである. PDPでは, ポリシーを判断するためのADIを入手しルール判定を下す. そして, ポリシーにマッチしたSQLを選別し, そのSQLを使ってDBにアクセスする. SQLでは, カラムレベルの制御はもとより, WHERE文などを使用することでレコードレベルの制御も可能である.“ 条件 ”として, 情報主体の属性情報を必要とする場合, SQLで, その条件に適合するデータのみを取得するといった制御ができる. このモデルでのアクセス制御は, ユーザのリクエスト単位で行われ, 戻りのデータ単位で行われるわけではない. 従って許可判断に要するオーバーヘッドは小さい.

このモデルの実現方法としては, 図2～図4で示したモデルで実装する方法が考えられるが, 別の方法として, PDPをフロントエンドに分離する方法を新たに提示する.（**図5**）この場合, フロントエンドで入手可能なADIから許可判断を行い, その判断結果をアプリケーション側のPEPに渡す. PEPは判断結果に基づいて使用するSQLを選択する. アプリケーションでは判断結果に基づくSQLの制御が必要になるが, PDPを切り離すことができるため, PDPの実装およびポリシーの管理は実施せずに済む. この方法が実現できれば, 実装および運用のコストを低減させることができる.

そのような利点を踏まえ, 本論文では, TAMebを用いることにより, このモデル2-1を実現するための具体的方法を提示する.

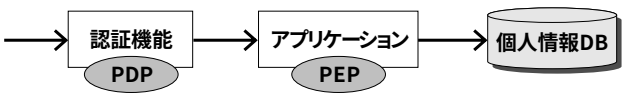

図5. モデル2-1. PDPをフロントエンドに分離するモデル

3.3 各アクセス制御モデルの比較

本章のまとめとして, 3.1節, 3.2節で述べた各アクセス制御モデルの特徴を**表2**に整理する.

4. 提案モデルを実現するための詳細モデル

本論文での提案モデルであるモデル2-1を実現するための詳細モデルを**図6**に示す. アプリケーション毎にEPALのルールセットを用意しておき, ユーザが, あるアプリケーションにアクセスする際, PDPは取得したADIから認可判断を行う. その後, 適合したルールと1対1に対応するロール情報をアプリケーションに渡す. なお, ロール情報とはユーザの権限を示した識別情報である. 次に, アプリケーション側では取得したロール情報をもとに, 使用するSQLを選別する. ここで, ルールとロールおよびSQLはそれぞれ1対1にマッピングされなくてはならない. アプリケーション側

表2.　各アクセスモデルの比較

		モデル1-1 “ アプリケーションでPDP，PEPを実施するモデル ”	モデル1-2 “ PDPをアプリケーションから分離するモデル ”	モデル1-3 “ PDPとPEPをアプリケーションから分離するモデル ”	モデル2-1 “ PDPをフロントエンドに分離するモデル ”
1	SQLのコントロールの可否	実現可能である.	同左	実現可能である. ただし，許可判断に必要なADIをPDPに渡すための仕組みが必要となる.	実現可能である. ただし，要件や既存システムの制約に対応できるかどうかを精査する必要がある.
2	SQLの結果のコントロールの可否	実現可能である.	同左	実現可能である. ただし，許可判断に必要なADIを取得するための仕組みが必要となる.	要件にもよるが，PDPがフロントエンドに存在するため，実現は困難である.
3	ポリシー管理の容易性	アプリケーション毎に管理する必要があり煩雑である.	一元管理できるため，ポリシーの管理は容易である.	同左	同左
4	アプリケーションの開発コスト	アプリケーション毎にPDP，PEPを実装する必要があり，既存システムに組み込む場合には，改修に伴う影響が大きい.	PEPはアプリケーション毎に実装する必要があるが，PDPは基盤コンポーネントとして切り出すことができる. 従って既存システムに組み込む場合には，モデル1-1に比べ，改修に伴う影響を抑えることができる.	PDP，PEPは基盤コンポーネントとして切り出すことができるが，ADIはアプリケーションなどから渡してもらう必要がある. しかしながら，複雑なロジックは必要ないため，既存システムに組み込む場合には，改修に伴う影響を極小化できる.	PDPは基盤コンポーネントとして切り出すことができるが，PEPはアプリケーション毎に実装する必要がある. しかしながら、複雑なロジックは必要ないため，既存システムに組み込む場合には，改修に伴う影響を極小化できる.

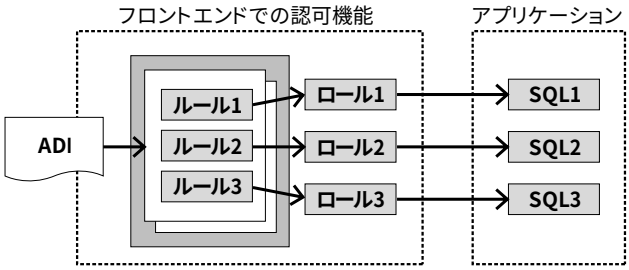

図6.　モデル1-2を実現するための詳細モデル

では，複数のSQLを用意し，ロール情報に応じてどのSQLを実行するかを識別する機能のみを実装すれば良い.

ADIは，前述したように，EPALルール要素である，“ ユーザカテゴリ ”,“ データカテゴリ ”,“ 目的 ”,“ 条件 ” に相当する情報である. まず,“ ユーザカテゴリ ”は，認証情報から取得可能である. 次に,“ データカテゴリ ”は，ユーザがブラウザから明示的に指定して送出する場合と，ブラウザからは送出されない場合がある. 前者は，ブラウザから送られてくる個人情報識別を取得し，ルールの判断情報として使用しなければならない. 後者の場合，アプリケーション側で，どの個人情報を参照するかを指定したSQLが用意されていれば良い.“ 目的 ”も同様に，ブラウザから明示的に指定して送出される場合と，ブラウザからは送出されない場合がある. 前者は，ブラウザからPOSTデータなどの形で送出される場合である. この情報はルール判断として使用されなければならない. 後者は，ユーザにとってアプリケーションにアクセスする行為そのものが目的に相当する場合である. この場合,“ 目的 ” はURLとマッピングできる.“ 条件 ”についても，PDPで取得可能なユーザの属性情報については，ルール判定の材料とする. 情報主体の属性情報などの，PDPで取得できないものは，SQLの中でWHERE文を使用することにより，その条件に適合したデータのみを取得するように制御する.

5. TAMebによる提案モデルの実現方法

TAMebのTag-ValueとV5.1から新しくサポートされたダイナミックルールエンジンを活用することで図6のモデルを実現する方法を示す. まず，5.1節と5.2節で図6の提案モデルをTAMebで実現するための，Tag-Valueとダイナミックルールエンジンの機能概要を説明する. 5.3節と5.4節で，ユーザIDから，ロールがアプリケーション毎に一意に決まるケースと決まらないケースに分けて，提案モデルの実現方法を提示する. 5.5節では，アクセス制御情報である，ロールを管理する機能の必要性について述べる.

5.1　Tag-Value

LDAP（Lightweight Directory Access Protocol）サーバは，ディレクトリサービスを提供するサーバであり，ユーザ情報などを管理することができる. TAMebのTag-Value機能を使用することにより，LDAPサーバに格納された任意のユーザ属性情報をHTTPヘッダに挿入してジャンクション経由でアプリケーションに送出することができる［5］［6］.

5.2　ダイナミックルールエンジン

ダイナミックルールエンジンとは，動的なアクセス制御を可能とするルールベースのアクセス制御機能である［10］［11］. ルールにより許可されるとアプリケーションへのアクセスが可能となるが，許可されない場合には，2種類の動作が選択できる. 一つはアプリケーションへのアクセスを禁止するという動作である. もう一つは，アクセスは禁止せず，TAMebがバックエンドのアプリケーションに対して，TAMebで独自定義されたAM_AZN_FAILUREというHTTPヘッダ

に, 予め設定しておいた文字列を挿入して送信するという動作である. このときアプリケーションは, 受け取ったAM_AZN_FAILUREヘッダ情報をトリガーにして固有の処理を実施することができる. (**図7**)

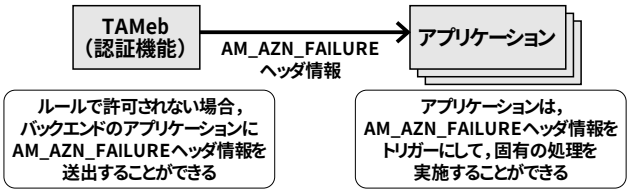

図7. ダイナミックルールエンジンの機能

なお, ダイナミックルールエンジンの許可判定に使用可能なADIとしては, まずユーザクレデンシャル内に格納されているユーザ名, グループ名, 拡張属性などがある. 拡張属性には, 例えばLDAPサーバから取得した, そのユーザの任意属性情報などがある. 次に, クライアントからのHTTPリクエストの中の情報を取り出して, それをADIとして使用することができる. 指定した任意のHTTPヘッダの値や, QueryStringの任意のコンテナ値や, POSTリクエストの任意のコンテナ値を取り出して, それをADIとして利用することができる. なお, ダイナミックルールエンジンはURL単位, つまりアプリケーション単位で個別に定義することができる.

5.3 ユーザIDから, ロールがアプリケーション毎に一意に決まる場合

ユーザIDから, ロールがアプリケーション毎に一意に決まるケースでは, アプリケーション毎に予め用意しておいたロール情報をアプリケーションに渡せば良い. 例えば, あるアプリケーションに対して“ マーケティング部門は, お客様が同意する場合(Opt-in)には, お客様の年齢と購買履歴情報を, 購買傾向分析の目的で参照する. ”というルールが定義されている場合, マーケティング部門の社員がアクセスすると, このルールが適用されることになる. 参照可能なデータカテゴリは,“ 年齢 ”と“ 購買履歴情報 ”であり, 目的情報は“ 購買傾向分析 ”であるが, これらがブラウザから動的に送出されることのないアプリケーションであれば, ロール情報はユーザIDから一意に決まる. このような場合には, ロール情報を予めLDAPのユーザ属性情報に格納しておき, アプリケーションにアクセスした時点で, Tag-Value機能によりその情報を渡せば良い.

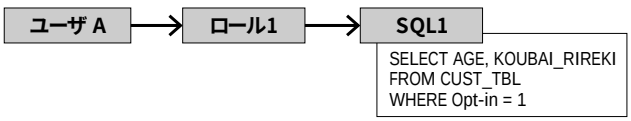

図8. ユーザとロールのひも付け(1対1のケース)

(**図8**) 条件に関しては,“ お客様が同意する場合(Opt-in)には ”という情報主体の属性情報に関する条件であれば, SQLの中にそれを選別するためのWHERE文を入れておけば良い.

5.4 ユーザIDから, ロールがアプリケーション毎に一意に決まらない場合

ユーザIDだけで, ロールがアプリケーション毎に一意に決まらないケースでは, PDPはADIとして渡される情報から, アプリケーションに渡すロールを動的に決めなければならない. 例えば, あるアプリケーションの参照ルールとして,“ マーケティング部門は, お客様が同意する場合(Opt-in)には, お客様の年齢と購買履歴情報を, 購買傾向分析の目的で参照する. ”というルールと,“ マーケティング部門は, お客様のメールアドレス情報を, 新サービス通知の目的で参照する. ”という, 同一部門に対して2つのルールが存在する場合, ADIの情報から動的にロールを選別して渡さなければならない. これを実現するには, ダイナミックルールエンジンのAM_AZN_FAILUREヘッダを利用する方法がある. つまり, 目的あるいは個人情報識別がブラウザから送出される場合, ダイナミックルールエンジンの判定において, 失敗したときにAM_AZN_FAILUREヘッダを送る機能を利用する. この場合, 5.3節のTag-Valueの結果とAM_AZN_ FAILUREヘッダの有無を組み合わせた情報をロール情報とする(**図9**). アプリケーションはこの組み合わせ情報を認識して, 使用するSQLを選択しなくてはならない.

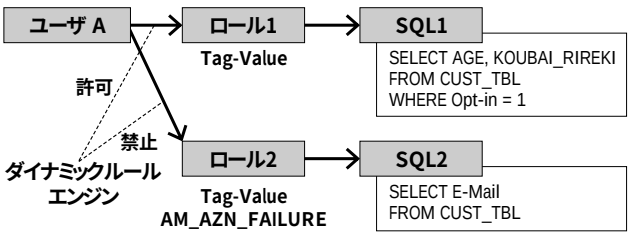

図9. ユーザとロールのひも付け(1対多のケース)

ここで, AM_AZN_FAILUREヘッダを活用する際の考慮点は, ルールセットに失敗したときしかAM_AZN_FAILUREヘッダの情報が送られないことである. ダイナミックルールエンジンでのルール判定の結果に応じて動的にAM_AZN_FAILUREヘッダのデータを替えることはできない. つまり, 同一ユーザに対して, アプリケーションに送出できるロールは2種類しか定義できない. もしも, この部分の機能が拡張されて, ルールの判断結果に応じて動的に切り替えることができれば, このTAMによる仕組みはより汎用的に適用できるようになる.

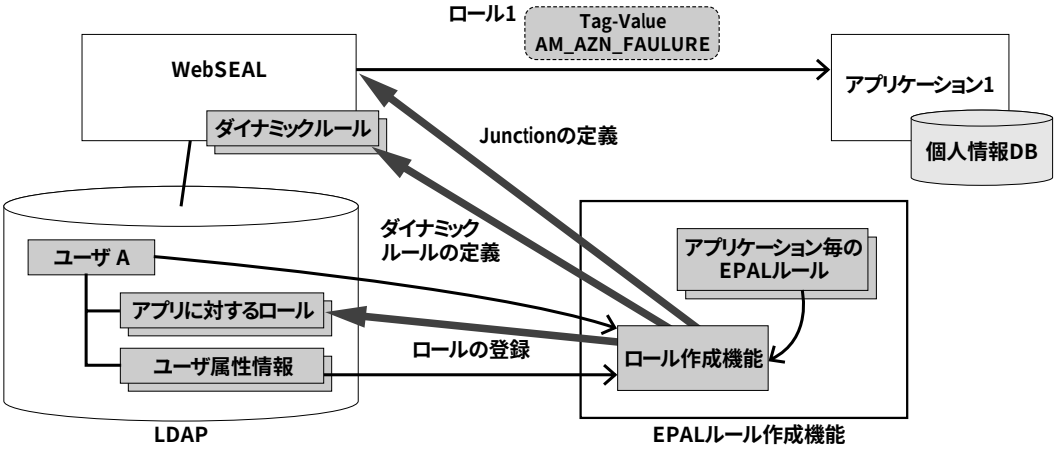

図10.　EPALルールに基づくロール作成機能

5.5　EPALルールに基づくロール作成機能

ロールを登録する際の作業では, ユーザ毎にLDAPのユーザ属性情報をアプリケーション単位で登録し, なおかつ必要に応じてダイナミックルールエンジンの定義をしなくてはならない. これをいちいち手動で行っていたら大変な作業になる. そこでEPALルールを作成すればロールが自動作成されるような“ロール作成機能”の実装を検討すべきである. アプリケーション毎にEPALルールを用意すれば, ユーザID情報, そしてユーザの属性情報をLDAPサーバから必要に応じて取得することで, 必要な情報の登録や定義を自動で実施する仕組みを構築すべきである(**図10**). なお, ロール名とアプリケーション側でのSQLは1対1に対応させなくてはならないため, EPALルールとロール名を決める際には, 予めアプリケーション担当者とアプリケーションの仕様について確認する必要がある.

6. おわりに

本論文では, 個人情報保護のためのアクセス制御システムに対して, 新たなモデルを提示し, 既存の製品であるTAMebを使用した実現方法を示した. この方法は, 前述したTAMebのAM_AZN_FAILUREヘッダの制約により, 汎用的な仕組みとして活用するには必ずしも十分でない部分もあり, 要件によっては対応できない場合はある. しかしながら, 例えば, 既にTAMebが組み込まれたシステムであれば, PDPの部分をそのままTAMebに任せることができ, PEPの役割を担うアプリケーションのロジックは簡単なもので済むため, 比較的容易に実装でき, 開発コストを低減させることが期待できる.

アクセス制御要件として, 汎用的な仕組みが必要であれば, 前述したMonitoring Tool for JDBCのようなソリューションが有効である. これは柔軟なアクセス制御ルールを組み込むことができるとともに, PEPの機能としては, ロールに応じてSQLをコントロールすることも, SQLの結果をフィルタリングすることもできる.

システムの検討に際しては, アクセス制御要件やシステム上の制約を考慮して, 最適な方式を選定する必要がある.

参考文献

[1] 渡辺 芳明, 情報開示管理ソリューション―情報漏えい防止と情報有効活用の両立, ProVISION No.42, pp.51-55, 2004
[2] IBM Japan., "Tivoli Access Manager for e-business" http://www.ibm.com/jp/software/tivoli/products/access_mgr.html
[3] IBM Corp., "Tivoli Access Manager for e-business" http://www.ibm.com/software/tivoli/products/access-mgr-e-bus/
[4] 沼尾 雅之, ビジネスにおけるプライバシー保護技術, ProVISION No.42, pp.25-29, 2004
[5] Enterprise Privacy Authorization Language (EPAL 1.2), http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/#Overview　'04.08.29
[6] G_nter Karjoth, Enterprise Privacy Authorization Language, http://www.zurich.ibm.com/pdf/privacysummit/Karjoth.pdf
[7] 沼尾 雅之, プライバシー情報管理の統合ソリューションテクノロジー, http://www.ibm.com/jp/software/isw/handouts/pdf/m/m_10.pdf , 2005
[8] IBM Corp., Application Privacy Monitoring for JDBC, http://www.alphaworks.ibm.com/tech/apm4jdbc
[9] Yoshinobu Ishigaki, Masayuki Numao, "Identity/Access Management for Integration and Privacy Protection in eCRM System of X-Auto",

http://www.zurich.ibm.com/pdf/privacysummit/Numao.pdf
[10] IBM Tivoli Access Manager for e-business WebSEAL 管理者ガイド バージョン 5.1 SC88-9851-00
[11] IBM Tivoli Access Manager Base 管理者ガイド バージョン5.1 SC88-9852-00

日本アイ・ビー・エム
システムズ・エンジニアリング株式会社
アーキテクチャー・イノベイティブ・ソリューション
アーキテクチャー・デザイン
ITアーキテクト
河岡 忠広 Tadahiro Kawaoka

[プロフィール]
1994年、日本IBM入社．1995年に日本IBMシステムズ・エンジニアリングに出向し，AIX，インターネット関連の技術サポートを経て，セキュリティを専門にさまざまなプロジェクトに参画．現在，ITアーキテクトとして，セキュリティーに加え，システム基盤のアーキテクチャー策定案件などにも携わっている．
情報セキュリティーアドミニストレーター
kawaoka@jp.ibm.com

日本アイ・ビー・エム
システムズ・エンジニアリング株式会社
システム・インフラストラクチャー・ソリューション
システムズ・マネジメント
ITスペシャリスト
住田 敦 Atsushi Sumida

[プロフィール]
IBM Tivoli セキュリティー製品の中で，Web/Webサービスに関する製品を中心に技術サポートを担当．
CISSP（Certified Information Systems Security Professional）
sumida@jp.ibm.com