情報統合実現のためのWebサービス要求に対するセキュリティー管理

川向 智之　住田 敦

Security Management for Web Service Requests to Realize Information Integration

Tomoyuki Kawamukou　Atsushi Sumida

メインフレーム上のアプリケーションが基幹データだけでなく, Webサービスなど多用なデータソースにアクセスする情報統合が着目されてきている. 本論文では, メインフレーム上のアプリケーションからDB2® for z/OS®を介してWebサービス要求を行う際のセキュリティー要件の実現法について検討する. 通信上必要となるセキュリティー要件を実現するためには, DB2と他のミドルウェアと組み合わせたシステムデザインが効果的であり, 具体化案として, Tivoli®製品と組み合わせた解決方法を提案する. さらに将来, 製品のセキュリティーが強化された場合のシステムデザインについても展望する.

Information integration is coming under the spotlight in which mainframe applications access not only mission-critical data but also various data sources such as Web Services. In this paper, we discuss how to realize the security requirements for Web Service requests invoked from applications on the mainframe via DB2® for z/OS®. To achieve the security requirements for communication, it is effective to combine DB2 and other middleware, and so, we propose solutions by using DB2 and Tivoli® security products. Moreover, we discuss system design by considering future security enhancements.

Key Words & Phrases：情報統合, 統合データソース, Webサービス, セキュリティー, 統合ID管理
information integration, integrated data source, web service, security, ID integration

1. はじめに

SOA（Service Oriented Architecture）に対する取り組みにおいて, 長年にわたり蓄積されてきたメインフレーム上の既存資産, 例えばIMS™（Information Management System）や, CICS®（Customer Information Control System）といったアプリケーションをWebサービス化し, オープンな環境へと公開する動きが高まっている. その一方で, メインフレーム上で稼動するアプリケーションが, 既存の基幹データだけではなく, Webサービスに代表される多様なデータソースにアクセスするというアプローチも考えられ始めている.

このアプローチの利点は, 既存データと外部Webサービスを同じインターフェースでアクセスする情報統合（Information Integration）を促進し, Webサービス化する以外にも, 既存のメインフレームアプリケーションに対して新たな価値を付加することが可能になるという点である［1-2］. そして, 情報統合実現のためには, さまざまなデータソースを単一のデータソースとしてアプリケーションからアクセスできる"統合データソース"が必要となる.

そこで, メインフレームにおける統合データソースの具体的な実現例としてDB2 for z/OS（以下, DB2と略す）を考える. DB2では外部のWebサービスにアクセスするための機能として, UDF（User Defined Function）を提供しており, SQLからこのUDFを呼び出すことで, Webサービスの提供するリアルタイムなデータを取得することが可能となる. メインフレーム上のアプリケーションからすると, SQLという従来のインターフェースを変えることなく, 既存資産であるDB2上の表とインターネット上のさまざまなサービスを同時に利用できることになる. このような「統合データソース」としての役割をDB2が担うことで, 今後, 堅牢（けんろう）なメインフレーム上のDB2を介したインターネット上へのアクセスが増加することが予測できる. ここで重要となるのが, 通信上のセキュリティー要件を満たすことである.

本論文では, 通信におけるセキュリティーという観

提出日：2006年8月31日　　再提出日：2006年12月18日

点からDB2のWebサービス要求の実現性を検討した（**図1**）.

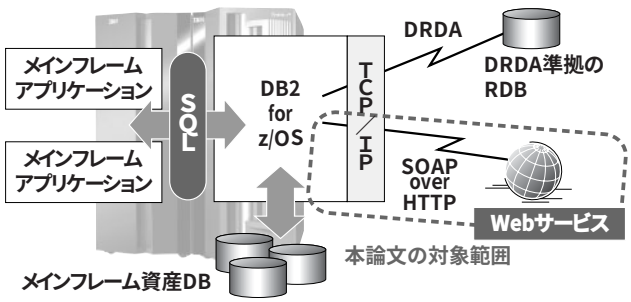

図1.　DB2から他データソースへのアクセス

以下、2章で, DB2からWebサービスを利用する場合の基本的な実装を紹介し, 3章で, その実装におけるセキュリティー上の課題を整理した上で, 4章で, その課題に対するTivoli製品との組み合わせによる解決方法を提案する. さらに, 5章では, 将来, DB2のUDFのセキュリティー機能が強化された場合のシステムデザインを展望する.

2. DB2におけるWebサービスの実装

本章では, まずDB2からどのようにWebサービスを利用するかをまとめる. 本論文では執筆時点で最新のDB2 for z/OS Version 8を前提とする.

2.1　DB2が持つ2つの機能的側面

図2のように, DB2はWebサービスに関して2つの機能を実装している［3-6］. 1つはWebサービス提供側としての機能であり, もう1つはWebサービス利用側としての機能である. 前者はWebSphere Application Server（以下, WAS）を介して, DB2へのSQL発行やストアードプロシージャのコールをWebサービスにラッピングする機能である. 後者がUDFを使用して外部Webサービスへの要求を行う機能であり, 本論文は情報統合という観点からこちらを対象とする. この場合, WASは必要とせず, DB2自身がWebサービスのリクエスターとなってメインフレームの外部との通信を行う.

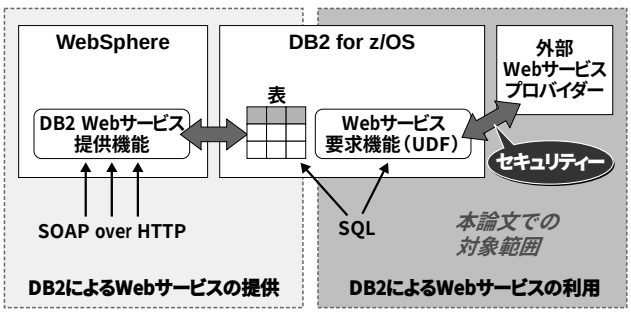

図2.　DB2が実装する2つのWebサービス機能

2.2　Webサービス要求を利用することの利点

メインフレームから他のデータソースにアクセスをする際に, Webサービスを基盤とした通信を採用する場合の利点はWSDL（Web Services Description Language）が使用できることである.

なぜなら, メインフレーム側の開発者は他のデータソースを持つシステム側から提供されたWSDLのXML（eXtensible Markup Language）文書を使用することによって, 接続性の保証されたインターフェースを生成することができるためである. さらに開発ツールを利用すれば, インターフェースの自動生成による開発容易性の向上だけではなく, 接続先のデータソースが増えた際の開発の迅速性も高めることができる.

2.3　UDFの作成と使用方法

ここで既存のメインフレームアプリケーションがSQLを用いて, どのようにWebサービスを使用するかについて簡単に説明する.

まずWebサービス利用のためのUDFをセットアップしておく. OS, DB2側に必要な前提条件をはじめ, UDFのセットアップに関する手順と環境設定については参考文献［7-8］を参照されたい.

図3はWebサービスへの要求で得たデータをDB2の表にINSERTする例である（DB2のマニュアル［8］から引用）. アプリケーションからSQLを発行する際, WebサービスUDFを呼び出し, 必要なパラメータを渡す. UDFはこの入力パラメータからSOAP（Simple Object Access Protocol）要求を組み立て, HTTP（Hypertext Transfer Protocol）でWebサービスにアクセスし, データを入手する. この例ではデータをその

```
事前準備:DB2提供のJCLでWebサービスUDFを作成

CREATE FUNCTION DB2XML.SOAPHTTPC (
          ENDPOINT_URL VARCHAR(256),
          SOAP_ACTION VARCHAR(256),
          SOAP_BODY VARCHAR(32672))
    RETURNS CLOB(1M)
      （以下省略）

使用時:SQL発行の際にUDFのパラメータに値をセット

INSERT INTO MYTABLE(XMLCOL)
VALUES (DB2XML.SOAPHTTPC(
'http://www.myserver.com/services/db2sample/list.dadx/SOAP',    ①
'http://tempuri.org/db2sample/list.dadx',                       ②
'<listDepartments xmlns="http://tempuri.org/db2sample/listdadx">
<deptno>A00</deptno>
</ListDepartments>'                                             ③
))
①：ENDPOINT_URL    ⇒  WSDLから入手
②：SOAP_ACTION     ⇒  WSDLから入手
③：SOAP_BODY       ⇒  ユーザーがXMLで記述
```

図3.　WebサービスUDFの使用例［8］

ままINSERT文のVALUE句の中に返し, DB2はそれを表の中に格納する.

図中, ①のENDPOINT_URLがアクセス先Webサービスの URL（Uniform Resource Locator）である. この例ではSQL発行の度にUDFに対してWSDLから得たURL情報をセットしている.

2.4 DB2から外部への通信方法の比較

DB2を通して分散システム上のリレーショナルデータベース（以後, 省略してRDBと表記）にアクセスする方法としては, DRDAを利用した方法が一般的である. DRDA®とはThe Open Group [9]で標準規格として採用されている, RDBの間の通信（TCP/IP, SNA：Systems Network Architecture）のためのアーキテクチャである. メインフレームアプリケーションからSQLを受け取ったDB2が, DRDAアプリケーション・リクエスターとしてリモートのRDBへとアクセスする.

一方でDB2からのWebサービス要求の場合, 外部データソースとの通信にDRDAは使用せず, 代わりにSOAP over HTTPを用いて行う. このリモートのデータソースに対する新しいアクセス形態については今後取り組むべき課題が多いと考えられるが, そのうちの1つが本論文のテーマでもあるセキュリティーである. DRDAおよびSOAP over HTTP, いずれもTCP/IPの上位レイヤーに位置するプロトコルであり, メインフレームアプリケーションから外部データソースへアクセスするための経路であることから, セキュリティーに関する検討は不可欠と考える.

3. 通信におけるセキュリティーの考慮点

DB2からWebサービスを利用する場合のセキュリティー上の課題を明らかにするため, DRDAが実現している機能と比較して考える.

3.1 DRDAアクセスにおけるセキュリティー実装

DB2がDRDAを利用して他のリモートのデータソースと通信を行う場合, DB2カタログ表の一部であるCDB（Communication Database）への情報の登録が必要になる（**図4**）. 例えば, 以下のSQLのように表の修飾子に"ロケーション名"を記述することで, リモートに存在するどのRDBMS管理の表であるかを識別することができる.

```
SELECT 列名 FROM ロケーション名.スキーマ名.表名
```

このロケーション名はCDB内でアクセス先RDBMSのIPアドレスやポート番号にマッピングされている.

また, SQLを発行したIDと, リモートのDBに対してアクセスする接続ID（アウトバウンドID）のマッピングもCDBに定義しておくことが可能である. このようにDRDAアクセスの場合はあて先情報およびIDについての一元管理が可能である.

さらにDRDAはそのプロトコルの中での暗号化が可能であり, 対象にはID, パスワード, および送受信される機密データが含まれる.

3.2 SOAP over HTTPアクセスの課題

WebサービスUDFの場合, リモートにあるデータソースのあて先情報はWSDLから入手したURLである. 前出の図3にもあるとおり, このURLはWebサービスUDFへの入力パラメータとして渡すことになる. SQL実行の度にパラメータを渡す方法と, 予めWebサービスUDFにURLを組み込んだ形で別のUDFを作成しておくことも可能であるが, いずれの場合でもWebサービスのあて先情報に関しては個別に管理するしかなく, CDBのような一元管理の仕組みを実現する必要がある.

次にIDについて考える. SOAPでは送信についての規定がないため, HTTPレイヤーのレベルでは基本認証を用いた送信が考えられる（基本認証以外で, 例えばフォーム認証などはRFC（Request for Comments）で規定されておらずベンダーの独自実装であるため, 一般論としての議論からははずして考える）. Webサービス・レベルでのIDの送信は, 認証情報を統一フォーマットで交換することが可能なOASIS [10]で策定された規格の1つである, SAML（Security Assertion Markup Language）[10]の認証アサーションの中に入れて送信するといった方法が考えられる. ただし, これら2つの規格については, UDFが未対応である.

DRDAにおける通信上の暗号化に対応する機能は, HTTPS（HTTP over SSL: Secure Socket Layer）が対応するため, 本来はSOAP over HTTPSで実現しなければならない. しかしながら, HTTPSについては現時点ではUDFが未対応である.

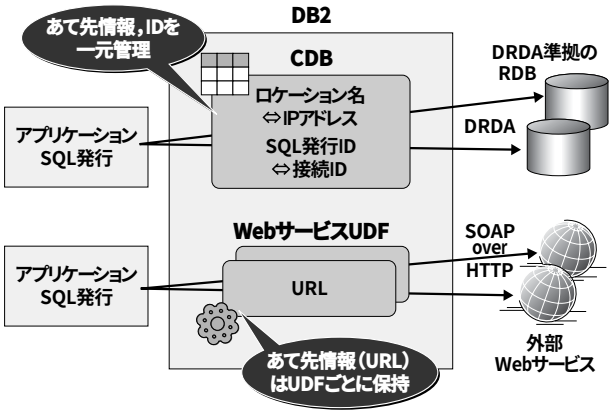

図4. DRDA接続とWebサービス要求の相違

このように, DB2がWebサービス・リクエスターとし

てメインフレームから外部への通信を行う場合，DRDAにおけるCDBと同等あるいはそれ以上のセキュリティー要件を満たす仕組みが求められる．両者において3つのセキュリティー要件への対応をまとめたものが**表1**である．

表1.　DRDAとDB2 Webサービス要求のセキュリティー機能の比較

接続形態（使用プロトコル）	あて先管理	ID管理	暗号化
DRDA接続（DRDA）	可	可	可
DB2 Webサービス要求（SOAP over HTTP）	不可	不可	不可

4．セキュリティー製品を使用したUDFによるWebサービスの保護

前章では，現在のUDFの機能でセキュリティー要件を満たすことが難しいことをまとめた．そこで，本章では他の製品と組み合わせることでDRDAと同等のセキュリティー強度を実現できるかを検討する．

4.1　セキュリティー要件ごとの製品検討

本節ではまず，前章で取り上げた3つの要件を満たすために必要となるセキュリティー製品を検討する．

a）あて先管理

あて先管理について，前章ではDRDAやUDFに対して任意のあて先を指定できるか否かという"あて先指定"の観点から検討を実施した．しかしながらあて先管理には指定したあて先に対してアクセスが可能であるか否かという"アクセス制御"の観点も重要であるため，ここでは2つの観点について検討する．

・あて先指定

3.2節で述べたように，あて先情報（URL）はUDFごとに保持しており，あて先の"指定箇所"について一元管理をするような製品は提供されていない．そのため，意図しないあて先が指定された場合に備え，次のアクセス制御での対応を検討する．

・アクセス制御

UDFを用いたシステムに対してアクセス制御機能を付加できるアプリケーションは大きく2つに分けられる．1つはファイヤーウォールで，もう1つはプロキシ製品である．

ファイヤーウォールをUDFのシステムに組み込むことにより，リクエスト元IPアドレスおよびリクエスト先IPアドレス，リクエスト先ポート番号といった情報を基にして，あて先に対するリクエストを許可する/禁止するといったアクセス制御を実装することができる．

これに対し，プロキシ製品を組み合わせれば，IPアドレスでのアクセス制御はもちろんのこと，リクエスト元のユーザー情報を送信することが可能となれば，ユーザーID単位でのアクセス制御が可能となる．また，本論で取り上げているようなHTTPベースの通信の場合，URL単位でのアクセス制御も可能となる．

b）ID管理

UDFからのWebサービス要求でユーザー情報をあて先のシステムへ送信するためには，プロキシ製品を使用し，Webサービス要求の送信元IPアドレスにマッピングされたユーザー情報をHTTPヘッダに埋め込むといった実装を行う．

c）暗号化

通信経路の暗号化に関しては，例えばVPNを用いてTCP/IPのレイヤーでの暗号化は可能である．また，HTTPのレイヤーでの暗号化であれば，プロキシ製品を使うことでデータソースとプロキシ製品の間についてはHTTPSを用いることで実現可能である．
以上の検討結果を**表2**にまとめる．

表2.　セキュリティー要件とコンポーネントの対応

	対応コンポーネント
あて先管理（アクセス制御）	プロキシもしくはファイヤーウォール
ID管理	プロキシ
暗号化	プロキシもしくはVPN

以上のように，UDFと組み合わせることでセキュリティー強度を上げることのできるコンポーネントを検討した．結論として，DRDAと同等のセキュリティー強度は，プロキシ製品を用いることで対応できる．このような機能を持つプロキシ製品はさまざまなベンダーが提供しているが，本論文ではIBMの提供するセキュリティー製品の中でTivoli製品を取り上げて検討する．

4.2　Tivoli製品で実装可能なセキュリティー

ここでは，前節で挙げたセキュリティー強度をTivoli製品により実装可能であるかを検討することに加え，プロキシを用いたUDFに対するセキュリティー強化をより詳細に検討する．

UDFはSOAP over HTTP通信を基盤としているため，まずはこの通信を以下の2つのレイヤーに分けて，セキュリティーを考える．

・HTTP通信については，HTTPプロキシなどによる認証およびアクセス制御
・SOAP通信については，Webサービス・セキュリティーでの保護

これらの機能をTivoli製品では以下のプロキシ機能を基にした製品で提供している．

- 認証プロキシ製品であるIBM Tivoli Access Manager for e-business（以後，省略してTAMと表記）
- TAMを機能拡張したWebサービス・セキュリティー製品であるIBM Tivoli Federated Identity Manager（以後，省略してFIMと表記）

つまり，Tivoliが提供するプロキシ製品によってHTTPレイヤーだけではなく，その上のレイヤーのSOAP通信も保護することが可能となる．

TAMではHTTPレベルでのさまざまな認証方式をサポートしている［11］．基本認証，Form認証だけではなく，HTTPヘッダ認証もサポートしている．HTTPS環境においてはクライアント証明書認証にも対応する．また，HTTPレベルだけではなく，IPアドレスによる認証も実施可能となっている．また，TAMは認証以外にも認証したデータを基に，あて先に対して任意のデータをHTTPヘッダに挿入して送信することが可能である．

これに対し，FIMはTAMのプロキシとしての機能をベースとして機能拡張をしたものであり，Webサービス・セキュリティーの中でも，SAMLといったようなドメインをまたいだシングル・サインオンや，分散システム間の信頼の保証，分散システム間での異なったID間のマッピングといった機能を提供する［12］．

したがって，まずDRDAと同等のセキュリティー強度を実装することを考えれば，TAMと組み合わせることで，少なくともIPアドレス認証によるあて先管理，HTTPヘッダへのユーザーID挿入によるID管理，HTTPSによる暗号化が可能となる．

5．DB2 UDFのセキュリティー機能強化を考慮したシステムデザインの展望

本章では，まず現時点のUDFの機能にTAMを組み合わせることで，さらにセキュリティー強度を高めるデザインを提案する．その後に，DB2のUDFが将来、基本認証とHTTPSに対応した場合と、さらにWebサービス・セキュリティーにまで対応した場合について、どのようなシステムデザインが可能になるかを展望する。

5.1　TAMが持つセキュリティー機能を用いた応用的なデザイン

まずTAMを使用した通信先に対する制御に関しては，アクセス可能か否かは，あて先IPアドレスだけではなく，URL単位の詳細レベルでの判断が可能である．

次に認証についてとなるが，現段階で採用可能な方法は，メインフレームとデータソース間をTAMによってIPアドレス認証させるという方式となる．

このIPアドレス認証には2つの観点がある．

1つ目は，TAMをメインフレーム側のシステムに配置することにより，メインフレームのIPアドレスを認証し，どのデータソース（データを保持するサーバ）に接続可能とするかというアクセス制御を実施するという観点である．

2つ目は，TAMをデータソース側に配置し，アクセスしてきたメインフレームIPアドレスを認証し，データソースを管理するシステム側で，どのデータソースに対してアクセスを許可するかを決定するという観点である．これはデータソースを管理する側できちんと制御するパターンであるが，IPアドレス認証を前提とする以上，NAT（Network Address Table）によってメインフレームのIPアドレスが分からなくなるようなケースではこのやり方は採用できない．

以上をまとめると，TAMのURLレベルでのアクセス制御をUDFにも適用可能であり，IPアドレスに基づくあて先管理については，TAMの位置によって2種類アクセス制御が可能となる．ただし，NATがある場合には現時点での採用に考慮点がある．これに対する解決策は次節にて検討する．

5.2　UDFが基本認証とHTTPSに対応後のシステムデザイン

今後，UDFが基本認証やHTTPSに対応した場合には，TAMとの間でのID管理，暗号化が可能となる．したがって，例えば，接続先のアプリーションがHTTPSに対応していない場合にもTAMとの間のみHTTPSにすることにより，少なくともこの経路だけは保護するといったデザインが可能となる．

また，前節にて取り上げたNATを使用した際の考慮点に関しても，基本認証を採用すれば，この考慮点だけが解消されるわけではなく，IPアドレスによるサーバ単位の認証から，データソースにアクセスしようとする（ユーザー）ID単位での粒度の細かな認証が可能となる．

ただし，基本認証に関しては，BASE64エンコードされたユーザーID・パスワードの文字列が通信経路上でリクエストの度に送信されることとなるため，ここでも UDFからTAMまでの間のHTTPS通信の採用が必須と考えるべきである．

このようにHTTPS通信が実装されれば，暗号化の要件についても対応可能となる．かつIPアドレスの偽装や，DNS（Domain Name Server）汚染による通信経路での偽装の対策の面からも，SSL相互認証を設定できることは非常に重要であるため，HTTPS対応の早期実装が望まれる．

5.3　UDFがWebサービス・セキュリティーに対応後のシステムデザイン

HTTPレベルでの認証や暗号化機能の実装だけで

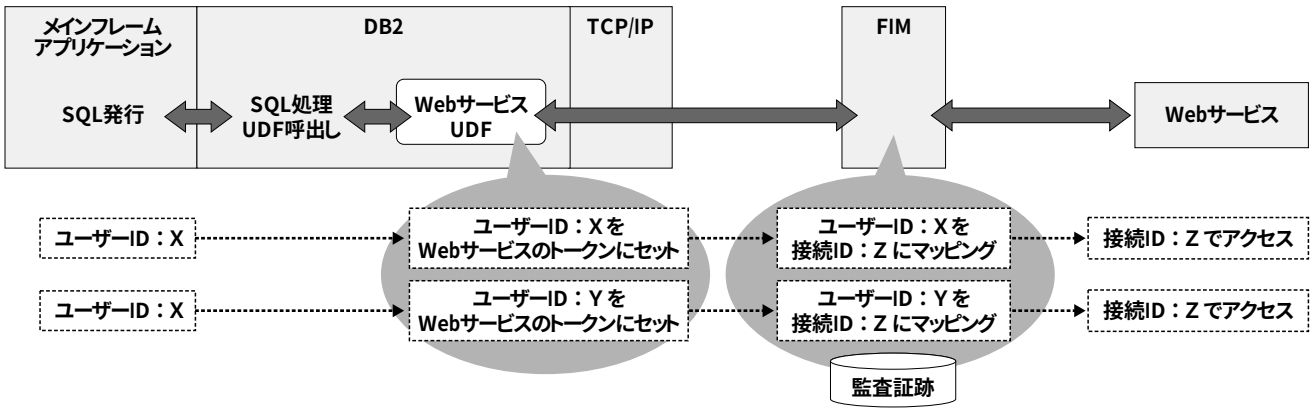

図5.　統合ID管理の将来像

はなく, Webサービス・セキュリティーに関してもUDFが機能を実装した場合にはどのようなデザインが可能となるかを次にまとめる.

まず, 基本的なWebサービス・セキュリティー上での認証の考え方であるが, SAMLなどの仕組みを用いた認証は, セッションが維持されている間のみ有効であり, そのセッションでのオペレーションに対する認証という意味が強い.

SAMLの認可アサーションを前提にしたアクセス制御を実施する場合, アクセス制御情報は集中管理されることとなるため, メインフレーム側から頻繁にIDを変えてデータソースにアクセスするような場合には, オペレーションごとの認可を確実に実施しやすいという利点がある.

また, SAML2.0ではアカウントリンクの機能を持っており, 分散システム上での異なるID間のマッピングを行い, それら異なるID間でのシングル・サインオンを実現させることができる. これは, SOAにおいて複数のサービスを統合する際にそれぞれが持つID情報の統合に関する解決策として紹介されているとおり, 既存の分散環境におけるさまざまなデータソースに対してメインフレームから接続する際に, それぞれのデータソースで使用するID情報をUDF自体が意識する必要がなくなることを意味する.

一方で, データソースに接続する際のIDは, アプリケーション単位程度に集約されているため,“ 誰が実際にデータソースにアクセスしたのか ”といったエンドユーザーのIDとのマッピングを行うのであれば, ロギングに注意しなければならない.

例えば, イントラネットの環境であってもインターネットの環境であっても, 日本版SOX法の施行にあわせ, システム利用時の監査を厳密に行わなければならなくなるケースが今後増えてくる. そうした際に, IDが途中で集約されてしまうと, IDとオペレーションのマッピング情報が失われることにより, 責任の所在が不明確になってしまう可能性がある. もちろん, メインフレーム側のログとデータソース側のログを追いかけて, エンドユーザーのアクションからデータソースへアクセスするオペレーションまでのフローをすべて解析するという方法も考えられなくはないが, トランザクションの多いシステムでは現実的な解決策ではない.

このような場合にも, FIMを前提としたWebサービス・セキュリティーによって対応は可能である. 例えば**図5**のようにデータソース側にアクセスする場合には単一のIDを用いるが, FIMのログにはエンドユーザーの実際のIDを記録するといった動作ができる. これにより, メインフレーム側とデータソース側のログのマージをして分析するという手間をかけずに, FIMによって記録されたログを基に追跡し, どのユーザーがどういったオペレーションを分散システムの中で行ったかを分析することが可能となる.

6. おわりに

メインフレームの資産とインターネット上のWebサービスを対象とした情報統合の実現には, DB2からのWebサービス要求という, 新たな通信手法を採用することになる. 従来のDRDA接続と比較して, 通信上のセキュリティーという観点から以下の3つの考慮点を指摘した.

(a)Webサービスあて先の管理
(b)Webサービスに関わるIDの管理
(c)通信上の暗号化

それぞれについて, Tivoliセキュリティー製品であるTAMを用いた以下の解決方法を提案した.

- アクセス制御機能による(a)の解決
- 基本認証による(b)の解決
- HTTPSによる(c)の解決

また, (a)の解決策に関しては, さらに細かくURLという単位でのアクセス制御が可能となることも述べた.

さらに, FIMによる統合ID管理のシステム構成を提案し, FIMを用いることによってドメインをまたいだリ

クエストのやり取りに関してもセキュリティーを確保し, かつIDに関する監査証跡の取得という観点から日本版SOX法へ対応した運用も可能となることも示した.

このように, DB2からのWebサービス要求にTivoli製品を組み合わせることで通信上のセキュリティーを高め, メインフレーム上における堅牢(けんろう)な情報統合の実現が可能となる.

なお, DB2のWebサービス要求UDFの今後の拡張に関しては, SAMLのようなWebサービス上の規格がバージョン・アップを繰り返しており(現時点で1.0から1.1を経て2.0にバージョン・アップをしている), どのバージョンを実装するかの判断が難しい面がある. したがって, まずHTTPレイヤーでのセキュリティー機能を拡充させ, その後にWebサービスに関連するセキュリティー機能の実装を検討していくという順序を取るべきだと考える.

謝辞

本論文執筆にあたり, 検討を行った各製品の技術担当者の皆様より, 多くの助言を頂きました. あらためて深謝いたします.

参考文献

[1] M.A.Roth, D.C.Wolfson, J.C.Kleewein, and C.J. Nelin: "Information Integration: A new generation of information technology," *IBM Systems Journal*, Vol.41, No.4, pp.563-577 (2002).
[2] S.Malaika, C.J.Nelin, R.Qu, B.Reinwald, and D.C. Wolfson: "DB2 and Web services," *IBM Systems Journal*, Vol.41, No.4, pp.666-685 (2002).
[3] Tim J Brown: "DB2 Web services for DB2 Practitioners," IBM White Paper (2004.05).
[4] Bart Steegmans et al.: "XML for DB2 Information Integration," IBM Redbook, SG24-6994 (2004).
[5] Dirk Wollscheid: "Building DB2 Web Service Consumer SQL Applications - Step by Step," IBM Data Management Technical Conference (2003.10).
[6] 日本IBM: "DB2 Information Integrator V8.1 ハンズオンセミナー Webサービス" (2003).
[7] DB2 UDB for z/OS Version 8 Installation Guide, GC18-7418-04.
[8] DB2 UDB for z/OS Version 8 Application Programming and SQL Guide, SC18-7415-03.
[9] The Open Group, http://www.opengroup.or.jp/ (2006.11).
[10] OASIS, http://www.oasis-open.org (2006.8).
[11] IBM Tivoli Access Manager for e-business 6.0 WebSEAL Administration Guide, SC32-1687-00.
[12] IBM Tivoli Federated Identity Manager 6.1 Single Sing-on Guide, GC32-0168-00.

**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
ビジネス・インフラストラクチャー・ソリューション
エンタープライズ・ミドルウェア
ITスペシャリスト**

川向 智之 Tomoyuki Kawamukou

[プロフィール]
**1999年, 日本IBMシステムズ・エンジニアリング入社.
DB2 for z/OSに関する技術サポートを担当し, お客様プロジェクトへの参画や製品情報の発信, 研修を実施.
kawamuko@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]
**2000年, 日本IBMシステムズ・エンジニアリング入社.
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**