XMLによるセキュリティ関連情報Webサービス

中村 章人　　戸村 哲
akihito@ni.aist.go.jp　s.tomura@aist.go.jp
産業技術総合研究所

情報セキュリティに対する脅威への対策として、平時からの情報収集と、システムの状態の把握及び頑健な状態の維持が重要である。しかし、情報量の膨大さと情報システムの多様性を考えると、これらを必要かつ十分に行うためのコストは大きく、適切な対策を実行するのは難しい。セキュリティ関連情報の収集や分析と、それらを用いた情報システムの管理とを相当程度自動化するシステムの開発・整備が求められている。

我々は、セキュリティ関連情報をプログラムで処理可能なXMLフォーマットで発信し、SOAPを使ってこれらの情報にアクセスするWebサービスを構築する。これによって、情報収集の自動化、検索や統計処理の高度化、アプリケーション間の連携等を可能にし、セキュリティ対策のプロセスを支援する。

本論文では、まずセキュリティ関連情報を提供するWebサービスの要件を整理し、具体的な脆弱性データベースを用いたWebサービスシステムの実現方法について述べる。また、セキュリティ関連情報の交換におけるメッセージ認証と発信者認証を、XML電子署名を用いて実現する方法を示す。

An XML-Based Security Information Web-Service

Akihito Nakamura　　Satoru Tomura
National Institute of Advanced Industrial Science and Technology (AIST)

For incresing the security of information systems, an intelligence gathering, an a priori determination of the system's current status, and keeping the system in a hardened state should be required. However, it is extremely difficult to complete these tasks and accurate solutions because of the huge amount of information to be published and great variety of the system components. It is expected to develop and deploy management systems that helps intelligence gathering and analysis, and maintenance of the information systems security.

To meet the need of security management, we provide a Web-service in which security-related infomation is published in computer-comprehensive XML format and exchanged by using SOAP. That enables us to realize an automatic intelligence gathering, advanced search and statistical analysis, and cooperation among applications and the security management process can be supported.

In this paper, we firstly discuss the requirements of a Web-service to provide security-related information. Then, we present an implementation of the Web-service using a vulnerability database. Also, message authentication and sender authentication methods, which are based on the XML digital signature specification, are discussed.

1 はじめに

情報セキュリティに対する脅威への対策として、国内外からの不正アクセス対策が重要性を増している。また、インターネットサーバやクライアントソフトウェアの脆弱性を攻撃するワームが広範かつ急速に広がることにより、情報システムが重大な影響を受けることが、NimdaやCodeRedにより実証された。

このような脅威に対抗するため、平時からの情報収集の重要性、及び緊急時に備えた情報収集体制とそのチャネルを維持することの重要性が改めて再認識されている。しかし、情報量の膨大さと情報システムの多様性を勘案すると、そのコストは大きい。また、情報収集を効率よく行えたとしても、システムの状態を正確に把握し、適切な処置を行うのは難しい[1]。従って、セキュリティ関連情報の収集や分析と、それらを用いた情報システムの管理とを相当程度自動化するシステムの開発・整備が重要である。

そこで我々は、セキュリティ関連情報をプログラムで処理可能な形式で発信することで、情報収集の自動化、検索や統計処理の高度化、アプリケーション間の連携等を可能にし、セキュリティ対策のプロセスを支援するシステムの研究開発を行っている。具体的には、セキュリティ関連情報の提供をWebサービス[11]として実現する方法を検討する。そして、Webサービスとそのアプリケーションを開発するためのツールキットを開発する。また、複数のWebサービスを統合する方法を検討する。ただし、現時点では、扱う情報をソフトウェア脆弱性に限定している。

Webサービスとは、メッセージとそのフォーマットにより仕様が規定されたネットワーク上で利用できるサービスで、そのメッセージを交換することによって実行される。Webサービスの要素技術であるSOAP[8]、UDDI[10]、WSDL[12]等はすべて標準化された技術で、XML[13]をベースにしているため、高い相互運用性と、実行環境からの独立性が期待で

きる。情報提供機能をWebサービスとして実現することで、これをアプリケーションを構成するコンポーネントとして利用できる。例えば、ネットワークから最新の情報を取得してホストのセキュリティ評価を自動的に行うツールの開発や、侵入検知システムのデータベースを動的に構成することが可能になる。

本論文では、まずセキュリティ関連情報を提供するWebサービスの要件を整理し、具体的なデータベースを用いてWebサービスを実現するシステムの実装について述べる。また、アプリケーションの開発を容易にするクラスライブラリと、発信情報のセキュリティを確保する方法について述べる。

本論文の構成は以下のとおりである。まず、第2章でセキュリティ関連の情報提供サービスで求められる要件を整理し、実現方法を検討する。第3章では、我々が利用するソフトウェア脆弱性情報について述べる。第4章では、具体的なデータベースを用いたWebサービスの実装について述べる。最後に、第5章でまとめと課題を述べる。

2 セキュリティ関連情報サービスの要件

以下に、セキュリティ関連情報サービスを実現するシステムに対して求められる要件を整理し、実現方法を検討する。

- データフォーマットに関する要件
  - データフォーマットが標準化されており、多様なアプリケーション間で相互運用性が確保されている。
  - データ構造の定義が可能である。
- アプリケーション開発・実行環境に関する要件
  - プラットフォームやプログラミング言語等に依らない、高い相互運用性が提供される。
  - ファイアウォールの内側にある計算機からサービスを利用できる。
- セキュリティに関する要件
  - メッセージを受信したとき、その作成者と内容の完全性とを検証できる(メッセージ認証)。
  - メッセージを受信したとき、それが誰から送られてきたかを検証できる(発信者認証)。

2.1 XMLによる情報交換

XML[13]は、プログラミング言語や実行システムやプラットフォームの制約を受けない、オープンな標準フォーマットである。データをXMLという標準フォーマットで表現することで、単一のデータを複数の目的に利用でき、アプリケーションの変更にも柔軟に対応できる。また、タグによるマークアップが自己記述的であることと、構造(文書型)の定義とその検証のフレームワークが用意されているため、プログラム処理に適している。さらに、個別に作られた情報を名前空間を用いて統合することや、高度な情報検索やナビゲーションをRDF[7]等のメタデータフレームワークを用いて実現できる可能性がある。

従来の分散システムでは、アプリケーションで共有・交換されるデータの構造を、プログラミング言語のデータ型で規定する。このため、分散アプリケーションの開発は自由に行えるが、プログラミング言語や実行システムが固有のものに限定される。一方、HTMLをデータフォーマットとするシステムでは、データ構造の定義ができないので、ブラウザ以外のアプリケーションの開発は困難である。

これらの観点から、我々は、セキュリティ関連情報の記述フォーマットとして、XMLを採用することにした。これは、個々の情報の記憶フォーマットをXMLに限定するものではなく、インターネットでの情報交換のフォーマットにXMLを採用し、またこれを推奨することで、セキュリティ関連情報のより効率的で効果的な利用を促進することを狙っている。

2.2 Webサービスのアーキテクチャとプロトコル

従来から分散システムの研究開発において、Webサービスと同様の概念が議論されている。Webサービスは、インターネットにより広く普及したHTTPプロトコル上に構築され、オープンなXML標準技術に基づいている点で、導入が容易で柔軟性が高い。また、プログラムではなくサービスをコンポーネントと考える点で抽象度が高く、サービスの動的な結合を可能にするアーキテクチャ[11]を持つ(図1)。Webサービスでは、サービスの公開、検索、利用のための情報交換を要求/応答メッセージの通信としてモデル化し、そのエンコーディングにはXMLを用いる。SOAP[8]は、W3Cで標準化されているWebサービスのメッセージ交換プロトコルである。

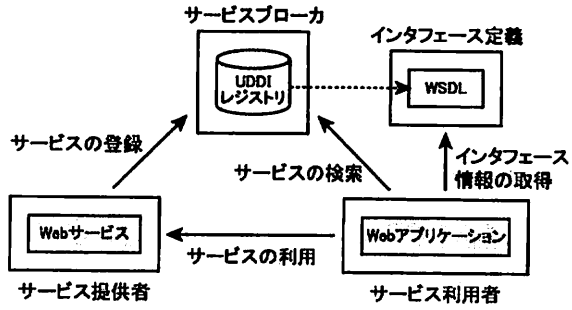

図1: Webサービスのアーキテクチャ

Webサービスのアーキテクチャの特徴は、我々が目指すシステムの開発・実行環境の要件を満たしているので、このアーキテクチャを利用する。つまり、SOAPのRPCモデルに基づいて情報提供サービスを実行する。しかし、Webブラウザを用いた情報表示の利便性も重要と考え、従来のHTTPを用いた情報提供サービスも共存させることにした。すなわち、Webアプリケーションに対して2種類の通信ポートを用意する(図2)。

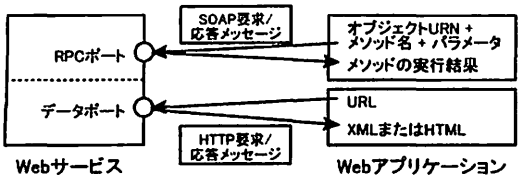

図 2: Web サービスの 2 種類のポート

RPC ポートを利用する Web アプリケーションは実行したいオブジェクトのメソッドとそのパラメータを SOAP 要求メッセージでサーバに送信し、サーバはメソッドの実行結果 (返値またはエラー) を SOAP 応答メッセージとして返す。SOAP メッセージの送受信は、実際には HTTP を用いて行うので、ファイアウォールの内側からでも Web サービスを利用できる。データポートを用いる Web アプリケーション (Web ブラウザ) は要求するデータの URL を HTTP の GET または POST メソッドで送信し、Web サービスは URL に対応するデータを返す。

2.3 メッセージ認証と発信者認証

セキュリティ関連情報のユーザの立場から、受信したセキュリティ情報を信頼する条件として、メッセージ認証と発信者認証が要求される。すなわち、メッセージの改ざん及びメッセージ作成の否認を回避できることを意味する。一方、提供者の立場では、通信内容の秘密性、すなわちメッセージの暗号化については考えない。扱う情報が一般公開を前提としているからである。同じ理由で、受信者の認証も考えない。また、情報の送信回数は問題にならないので、リプレイ攻撃は脅威と考えない。

CERT/CC のメールによるセキュリティ勧告通知サービスでは、このメッセージ認証と発信者認証を PGP の電子署名で実現している。我々の実現方法については、4.5 節で述べる。

3 標準化された脆弱性情報の利用

ここでは、我々が利用するソフトウェア脆弱性情報について述べる。

3.1 CVE 互換性

CVE (Common Vulnerabilities and Exposures)[3] は、公知のソフトウェア脆弱性を集積した「脆弱性の辞書」である。それぞれの脆弱性に一意な識別子を付与し、脆弱性の包括的なリストを作成し、これを業界標準として用いるという試みである。セキュリティ関連のコミュニティや米国政府関係の組織が CVE を利用し始めている。

CVE の各エントリはそれぞれ一つの脆弱性を表しており、以下の項目から構成される。

- **CVE 名:** 脆弱性の一意な識別子。
- **説明:** 脆弱性についての簡単な説明文章。
- **照会先リスト:** 詳細な情報を得るための情報源のリスト。

脆弱性を扱うツール、データベース、サービス等が CVE 名を参照することで、それぞれ独自に名前付けされた情報を相互に関連付けることができる。CVE 名を取り入れた製品やサービスを、「CVE 互換 (CVE-compatible) である」という。我々は CVE 互換な Web サービスを提供し、CVE 互換のサービスやツール同士を統合するための基盤を構築する。

3.2 ICAT Metabase の利用

ICAT Metabase[4] は、米国 NIST (National Institute of Standards and Technology) が提供する、CVE 互換の脆弱性データベースである (以下では ICAT と略す)。ICAT では、CVE の照会先から得られる情報を分析し、個々の脆弱性情報を 40 個程度のカラムで記述する (表 1)。表 1 で省略したカラムとして、攻撃による被害の種類、脆弱性が発生する OS の種類等がある。

我々は、ICAT を脆弱性情報の中心的なデータベースとして位置付け、これを Web サービスとして提供する。また、この Web サービスを利用してアプリケーションを開発するためのクラスライブラリを提供する。

4 脆弱性情報 Web サービスの実装

我々が実装した Web サービスと、アプリケーション開発用のクラスライブラリについて述べる。

4.1 システム構成

本システムの構成を図 3 に示す。SOAP プロセッサは、SOAP メッセージの送受信と Web サービスオブジェクトに対するメソッド起動を行う。Web サービスオブジェクトは、Web サービスを実行するオブジェクトである。リソースアダプタは、実際に用いる個々のデータ記憶システムの相違を吸収する。ここでは RDB を用いるが、ファイルシステムや LDAP を用いることもできる。

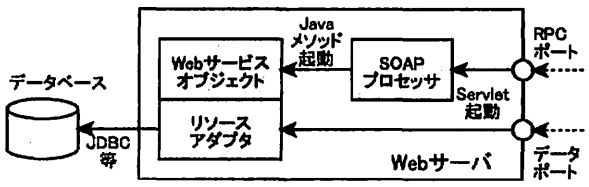

図 3: Web サービスの構成要素

本システムの実装には表 2 のソフトウェアを用いた。Linux と Windows の 2 種類の OS を対象とし、いずれの OS 環境でもシステムを構築できるものを選んだ。Apache SOAP を利用するためには Servlet コンテナが必要であり、ここでは Tomcat を用いた。プログラミング言語は、実行環境への依存性が少な

カラム名	説明
CVE_ID	CVE 名。
Publish_Date	情報の公表日。
Severity	被害の深刻度。High、Medium、Low のいずれか。
CVE_Description	脆弱性の説明。
Vuln_Software	脆弱なソフトウェアのリスト。
AR_Launch_remotely	攻撃の要件: ネットワークを介して遠隔から攻撃が可能である。
AR_Launch_locally	攻撃の要件: 攻撃対象のシステム上で直接攻撃が可能である。
AR_Target_access_attacker	攻撃の要件: 攻撃者が仕掛けたリソースに非攻撃者がアクセスする必要がある。
VT_Buffer_overflow	脆弱性の種類: システムへの入力が、想定していた長さを越える。
VT_Access_validation_error	脆弱性の種類: アクセス制御機構に不具合がある。
VT_Exceptional_condition_error	脆弱性の種類: 例外処理に不具合がある。
VT_Envionment_error	脆弱性の種類: システムがインストールされた環境に問題がある。
VT_Configuration_error	脆弱性の種類: ユーザの設定に問題がある。
HL1,...,HL5	より詳細な情報 (照会先) へのリンクを表す URL。
Link1Source,...,Link5Source	HL1,...,HL5 のリンク先のサイト名。CERT/CC や ISS X-Force 等。
Link1Name,...,Link5Name	Link1Source,...,Link5Source のサイトにおける固有の脆弱性識別名。
Link1Type,...,Link5Type	HL1,...,HL5 で提供される情報の種類 (General、Patch、Exploit)。

表 1: ICAT における脆弱性情報の記述内容 (一部省略)

く、XML 関連のツールやライブラリが数多く提供されている Java を用いた。

OS	Linux 2.4.19-34 または Windows 2000 SP2
Web サーバ	Apache HTTPD 1.3.26
SOAP プロセッサ	Apache SOAP 2.3.1
Servlet コンテナ	Apache Tomcat 4.0.4
データベース	MySQL 3.23.51-1
Java	J2SE 1.3.1_04

表 2: 実装に用いたソフトウェア

4.2 オブジェクトモデル

まず、アプリケーションから見た ICAT 及び Web サービスをモデル化する (図 4)。このモデルに基づいて、Java クラスライブラリ、XML の文書型定義、及び Web サービスのインタフェース定義 (すなわち WSDL[12]) を提供する。

我々はモデルの記述に UML を用いた。XML スキーマ言語を用いてもデータモデルを記述できるが、UML では制御や GUI のためのオブジェクトを含むシステム全体のモデルを実装言語に依存せずに記述できる利点がある。UML から XML 文書型や Java クラスへのマッピングを行うことでシステムの実装を進め、修正や改良は UML にフィードバックする。

以下に、主なクラスの役割を示す。

クラス `ICATEntry` とその部品クラス: ICAT エントリを表す。ここでは、同じ種類のカラムや、複数のカラムで一つのデータを表すものをまとめて、その単位でクラスとした (表の罫線で区切ったまとまり)。例えば、攻撃の要件を表す三つのカラムをまとめて一つのクラス `AttackerRequirements` にカプセル化し、その属性として三つのカラムを持たせる。

インタフェース `ICATMetabase`: ICAT のサービスを規定する。以下のメソッドを持つ (他にエントリ数や全 CVE 名を返すメソッドがある)。

- `ICATEntry getEntry(String cveID)`: CVE 名を指定してエントリを取得する。
- `SearchResult search(SearchCriteria sc)`: 指定した条件を満たすエントリのリストを返す。条件は `SearchCriteria` に、結果は `SearchResult` にカプセル化する。

クラス `ICATSQLMetabase` と `ICATJDOMetabase`: `ICATMetabase` の具象クラスで、Web サービスオブジェクトの役割を果たす。それぞれ JDBC と JDO[6] を利用して ICAT を記憶したデータベースにアクセスする。つまり、リソースアダプタの機能を実現する。他のデータ記憶システムを利用したい場合、同様の具象クラスを追加するだけで済み、他を修正する必要がない。

クラス `SOAPProxy` と `ICATSOAPProxy`: `SOAPProxy` は、SOAP RPC の実行機能をカプセル化した汎用的なプロキシで、SOAP メッセージの送受信を行うメソッド `invokeMethod` を持つ。これを拡張した `ICATSOAPProxy` は、`ICATMetabase` をインタフェースとして Web サービスにアクセスするためのプロキシである。このクラスを用いることで、SOAP の利用を意識せずにアプリケーションの開発を行える。

4.3 SOAP RPC の実行

RPC チャネルを用いた Web サービスの実行シーケンスを示す (図 5)。

1. アプリケーションオブジェクトは、まず `ICATSOAPProxy` オブジェクトを生成し、`ICATMetabase` で定義されたメソッド m (`getEntry` や `search` 等) を呼び出す (図 5 ①)。

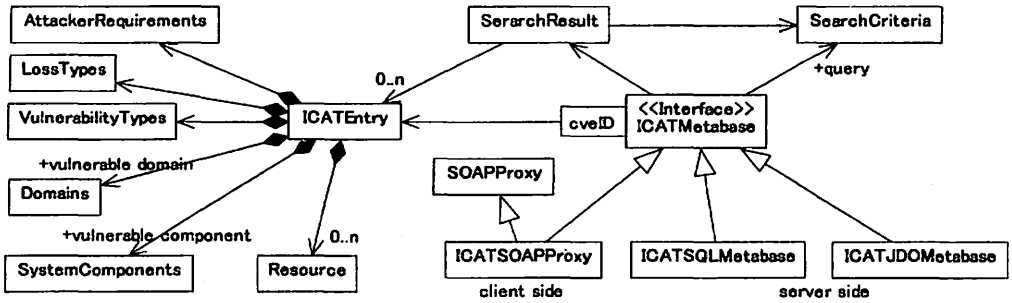

図 4: ICAT のオブジェクトモデル (UML クラス図)

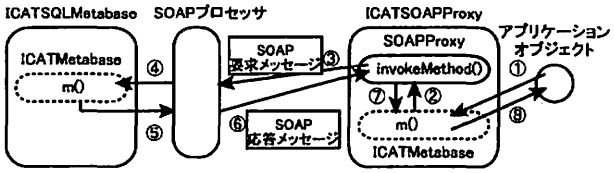

図 5: SOAP RPC の実行シーケンス

2. メソッド m は、SOAPProxy のメソッド invokeMethod を呼び出す (図 5 ②)。このとき、メソッド名 m と m 自身のパラメータを渡す。invokeMethod は、このメソッド名とパラメータ、Web サービスオブジェクトの URN を含む SOAP 要求メッセージを作成し、サーバに送信する (図 5 ③)。

3. サーバ側の SOAP プロセッサは、SOAP 要求メッセージを受信したら、URN で指定された Web サービスオブジェクトのメソッド m を呼び出す (図 5 ④)。

4. Web サービスオブジェクト (例えば ICATSQLMetabase のインスタンス) は、メソッド m を実行する。

5. SOAP プロセッサは、Web サービスオブジェクトからメソッドの返値を受け取り (図 5 ⑤)、これを SOAP 応答メッセージで返送する (図 5 ⑥)。

6. SOAPProxy のメソッド invokeMethod は SOAP 応答メッセージを受け取り、これを invokeMethod の返値とする (図 5 ⑦)。

7. ICATSOAPProxy のメソッド m は、invokeMethod の返値である SOAP 応答メッセージから m の返値であるオブジェクトを生成し、これをアプリケーションオブジェクトに返す (図 5 ⑧)。

4.4 SOAP メッセージのエンコーディング

SOAP RPC では、メソッドのパラメータ及び返値を XML にエンコードして要求/応答メッセージで送受信する。このためには、アプリケーションの型システムを SOAP の型システムにマッピングし、オブジェクトと XML との間で相互変換する必要がある。本システムでは、SOAP の仕様で推奨されているいわゆる Section5 エンコーディング [8, 5 節] を用いる。

我々が用いた Apache SOAP は、Section5 エンコーディングをサポートし、Java 用のエンコーダを提供している。ただし、Java の単純型と、配列及び一部のコレクションクラス以外のクラスは、JavaBeans 仕様 [5] に基づくクラスと見なして Bean のプロパティだけをエンコードする。このため、RPC のパラメータまたは返値となるクラス (ICATEntry とその集約クラス、SearchCriteria と SearchResult) を JavaBeans 仕様に従って実装した。つまり、ICAT エントリの各カラムは、Bean のプロパティとして get/set メソッドでアクセスできる。

本システムにおける SOAP メッセージの例を図 6 に示す。図 6(1) は、メソッド getEntry の要求メッセージで、パラメータの CVE 名は Java の String 型から SOAP の string 型 (要素 cveID) にマッピングされる。図 6(2) は、同メソッドの応答メッセージで、返値の ICATEntry オブジェクトがエンコードされている。例えば、Java の Date 型のプロパティ publishDate は SOAP の dateTime 型にマッピングされる。Java オブジェクトにデコードされた後は、Bean の get メソッド getPublishDate() でこのプロパティを読み出す。

4.5 XML 電子署名

既に述べたように、セキュリティ関連情報を提供する Web サービス自身のセキュリティを確保するために、メッセージ認証と発信者認証を実現する。発信者認証は Web システムで広く利用されている SSL を用いて、メッセージ認証は XML 電子署名 [14] 及び SOAP 電子署名拡張 [9] を用いて実現する。以下では、メッセージ認証の実装について述べる。

データポートから発信する XML 文書への署名は、XML 電子署名に従う。その実装として、IBM alphaWorks の XML Security Suite [2] を用いた。

図 7 に署名付きの XML 文書の例を示す。この例では、署名アルゴリズムに DSS (DSA) を用いている。要素 dsig:Object の子要素が署名対象のデータ、SignatureValue の内容テキストが署名、SignatureMethod が署名アルゴリズムを表す。

SOAP 電子署名拡張では、SOAP エンベロープで署名を運ぶための SOAP ヘッダの構文と処理規則を規定しているが、鍵や証明書の生成・管理については規定していない。署名に基づく信頼を構築するには、これらの手段が必要である。また、これを実装している SOAP プロセッサはない。このため、本シス

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="...">
 <SOAP-ENV:Body>
  <ns1:getEntry xmlns:ns1="urn:icat">
   <cveID xsi:type="xsd:string">
   CVE-2002-0005</cveID>
  </ns1:getEntry>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(1) メソッド getEntry の要求

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="...">
 <SOAP-ENV:Body>
  <ns1:getEntryResponse xmlns:ns1="urn:icat">
   <return xsi:type="ns1:ICATEntry">
    <cveID xsi:type="xsd:string">
    CVE-2002-0005</cveID>
    <publishDate xsi:type="xsd:dateTime">
    2002-01-30T15:00:00.000Z</publishDate>
    <attackerRequirements
    xsi:type="ns1:AttackerRequirements">
     <remoteLaunch xsi:type="xsd:boolean">
     true</remoteLaunch>
     <localLaunch xsi:type="xsd:boolean">
     false</localLaunch>
     <resourceAccess xsi:type="xsd:boolean">
     false</resourceAccess>
    </attackerRequirements>
    ...
   </return>
  </ns1:getEntryResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(2) メソッド getEntry の応答

図 6: SOAP メッセージの例 (一部省略)

テムでは、今のところ、SOAP 電子署名拡張に従ったメッセージ認証を実装していない。これに代わる方法として、独自のエンコーダを実装し、SOAP ボディにオブジェクトと署名の両方を含める方法を検討している。

5 おわりに

本論文では、XML をデータフォーマットに用いてインターネット上でセキュリティ関連情報を共有・交換する方法について述べた。SOAP RPC による Web サービスを実装するためのシステムアーキテクチャを設計し、具体的な情報として CVE 互換の脆弱性データベース ICAT Metabase を提供する Web サービスを実現した。また、公知のセキュリティ関連情報を交換する場合のセキュリティ要件を検討し、電子署名を用いて XML のメッセージ認証と発信者認証を実現した。

現在、(財) 国際情報化協力センターの事業の一環として、本システムの評価を行っている。今後、侵入検知システムやセキュリティチェックツールと本システムを統合したときの効果を、実証実験で確認していく計画である。

```
<Signature
xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo>
  ...
  <SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#dsa-sha1">
  </SignatureMethod>
   <Reference URI="#Res0">...</Reference>
 </SignedInfo>
 <SignatureValue>Mha...xYw==</SignatureValue>
 <KeyInfo> ... </KeyInfo>
 <dsig:Object xmlns=""
 xmlns:dsig="..." Id="Res0">
  <ICATEntry>
   <cveID>CVE-2002-0005</cveID>
   <publishDate>2002-01-30</publishDate>
   ...
  </ICATEntry>
 </dsig:Object>
</Signature>
```

図 7: 署名付き XML 文書の例

参考文献

[1] Arbaugh, W.A., et al.: Windows of Vulnerability: A Case Study Analysis, *IEEE Computer*, Vol.33, No.12, December 2000, pp.52-59.

[2] IBM alphaWorks: *XML Security Suite 20020422.*

[3] Martin, R.A.: Managing Vulnerabilities in Networked Systems, *IEEE Computer*, Vol.34, No.11, November 2001, pp.32-38.

[4] National Institute of Standards and Technology: *ICAT Metabase: A CVE Based Vulnerability Database.* http://icat.nist.gov/

[5] Sun Microsystems: *JavaBeans API Specification*, Version 1.01, July 1997.

[6] Sun Microsystems: *Java Data Objects.* http://access1.sun.com/jdo/

[7] W3C Recommendation: *Resource Description Framework (RDF) Model and Syntax Specification*, 22 February 1999.

[8] W3C Note: *Simple Object Access Protocol (SOAP) 1.1*, 08 May 2000.

[9] W3C Note: *SOAP Security Extensions: Digital Signature*, 06 February 2001.

[10] UDDI.org: *Universal Description, Discovery and Integration (UDDI).* http://www.uddi.org/

[11] Curbera, F., et al.: Unraveling the Web Services Web - An Introduction to SOAP, WSDL, and UDDI, *IEEE Internet Computing*, Vol.6, No.2, March/Aprial 2002, pp.86-93.

[12] W3C Note: *Web Services Description Language (WSDL) 1.1*, 15 Marsh 2001.

[13] W3C Recommendation: *Extensible Markup Language (XML) 1.0 (Second Edition)*, 6 October 2000.

[14] W3C Recommendation: *XML-Signature Syntax and Processing*, 12 February 2002.