# オブジェクト指向分散環境OZ++におけるインターネットセキュリティ

大西 雅夫*(東洋情報システム)　濱崎 陽一(電子技術総合研究所)
西岡 利博*(三菱総合研究所)　塚本 享治(電子技術総合研究所)
* 開放型基盤ソフトウェアつくば研究室研究員

近年、インターネット上での情報公開が一般的となっている。ソフトウェアもネットワーク上で共有することにより、再利用が進むという期待があり、ソフトウェアを自動配送するシステムが現れてきた。しかしこれには、悪意のあるソフトウェアからいかに重要な情報やシステムを守るか、という問題が伴っている。本稿では、ソフトウェアの共有と配送を基本とするOZ++を取り上げ、インターネットセキュリティの検討結果を報告する。

## 1　はじめに

WWWによる情報の共有が普及した現在、新たに、広域ネットワークを介してプログラムを共有する形態にも進歩が現れ始めている。従来はftpなどの手段でプログラムを入手し、インストールを行って実行していたが、新しい形態では、アプリケーションの実行が進む過程で、必要になったプログラムを自動的に次々取り寄せながら処理を継続してゆく。このことにより、開発者は相手側に必要なプログラムがあるかどうかを気にすることなくプログラムを再利用できるし、利用者はアプリケーションを利用するためにその都度インストールする手間をはぶくことができる。また、プログラムが送信された側で実行されるという特徴を活かして、アニメーションの交換のような新しいネットワークサービスの実現も可能となる。

しかし、必要に応じて配送されて来るプログラムが、何の制限もなしに実行されるとすると、組織の機密事項やプライバシーに関わる情報を読み出して勝手に外部に送信したり、システムを破壊する可能性があり、危険この上ない。

以下、先ず次節で、このような形態で求められるインターネットセキュリティ上の要件を検討する。次に3節で、そのような形態を実現するシステムの例として、我々が研究開発中のオブジェクト指向分散環境OZ++のセキュリティ上の特徴を示す。ここで、LAN上のマルチユーザ環境としてOZ++が備えるセキュリティが示される。4節で自動配送を経てプログラムが実行されるシステムに特有のセキュリティを検討する。5節で、ファイアウォールを介したインターネットでのOZ++の運用形態を、6節でその場合のセキュリティを説明する。

## 2　インターネットセキュリティ上の要件

一般に組織はセキュリティ方針に基づいてインターネットに接続する。セキュリティ方針とは何を許容範囲とし、それ以外のことにどのように対処するかを決めたものであり、当然、組織によって全く異なる[1], [2]。

前節で述べたように、LANや広域ネットワークを介したプログラムの配送を前提としてプログラムの共有化が進めば、様々なメリットが生じるが、一方でセキュリティ上の脅威も発生する。このような新しい形態のシステムを普及させるためには、先ず、利便性を可能な限り保ったままでそのシステム自身が持つべき最低限のセキュリティ上の要件を明確化し、組織が採用しているセキュリティ方針に沿った運用が可能かどうか判断できるようにすることが必要である。そこで、そのようなセキュリティ要件を次のように設定した。

- 既存の情報の流出や破壊が発生しないこと（既存環境のセキュリティを損なわないこと）。

---

Internet Security of OZ++: An Object-Oriented Distributed Systems Environment
Masao Onishi* (Toyo Information Systems, Co., Ltd.),
Yoichi Hamazaki (Electrotechnical Laboratory),
Toshihiro Nishioka* (Mitsubishi Research Institute),
Michiharu Tsukamoto (Electrotechnical Laboratory)
*: Researcher, Tsukuba Laboratory, Open Fundamental Software Technology Project

- 組織内部に新システムで蓄えられた情報や新システムで構築されたサービスが、権限のない外部の人間によって、アクセスされたり破壊されたりしないこと。
- 新システムの使用によって、外部の人間による組織内部の資源の不正利用が発生しないこと。

ただし、情報流出やサービスの破壊について、内部の人間の関与がある状況は考慮しないこととする。自分が自由にできる情報を外部に流出させるには、通信を利用する以外にも種々の方法があるため、システムで対処する意味がないからである。また、資源やサービスを利用不能にする使用不能攻撃にも対処しない。その対処は各サービスに依存し、一般的な対処は困難なためである。

以下、OZ++ではこれらの要件をどのように満たし得るかを3.3節、6節で述べる。

このような基本的な要件を満足させた上で、可能な限り多くのセキュリティ方針をカバーできるように、次の項目に関して利便性とセキュリティのトレードオフを調整できるようにする必要がある。

1. 外部からのアクセスを許容するか
2. 外部から配送されるプログラムの選択実行を行うか

OZ++における実現方針を、1については6.2節で、2については4.2節、7.2節で述べる。

## 3 OZ++のセキュリティ上の特徴

データとそれを処理するプログラムをひとかたまりと考えるオブジェクト指向パラダイムを用いれば、プログラムの共用を自然に表現できる。開発時には既存のクラスをネットワークを介して参照し、再利用すればよいし、実行時には渡って来たデータ(インスタンス)に対してメソッド呼び出しを行うと、そのメソッドを実装するプログラムが予めインストールされていなくても、自動的に配送されて実行されればよい。ネットワークに分散するサービスの利用も、オブジェクトを共有し、利用し合うことと考えることができる。OZ++はこのように、オブジェクトの交換と共有を可能にする環境であり、ネットワークサービスを含む様々なアプリケーションを開発、運用するためのプラットフォームでもある。

3.1節でプログラム配送のメカニズムを説明する。

また、OZ++では、既存環境や他言語プログラムを利用する手段を与えることで、効率のよいアプリケーション開発が行えるように配慮している。しかし、このことにより、セキュリティ上、いくつか問題が発生する。そこで、3.2節で、OZ++における他言語インタフェースと外部プロセスの利用について簡単に説明する。

LAN環境においてOZ++は、マルチユーザ環境であり、ユーザの権限に基づくアクセス制御によって保護される様々なアプリケーションが運用される環境である。3.3節で、LAN環境におけるセキュリティを説明する。

### 3.1 プログラムの配送の実現

分散サービスの内容やその相互運用を記述しやすいように、OZ++言語[3]が提供されており、OZ++プログラムはこの言語でクラスを単位として記述される。エグゼキュータは、OSの一プロセスであり、OZ++プログラムをマルチスレッドで実行するランタイムシステムである[4]。各エグゼキュータにはクラスオブジェクトが存在する。クラスオブジェクトはクラス生成時に全世界でユニークなクラスIDを付与し、そのクラスIDをキーとしてクラスを管理する。クラス配送とは、エグゼキュータやエグゼキュータ上で実行されるプログラミング環境などがクラスを必要とし、そのエグゼキュータのクラスオブジェクトがそれを持っていなかった場合に、そのクラスオブジェクトの要求に従って、そのクラスを所持するクラスオブジェクトから必要なオブジェクトやファイルが送られることである。エグゼキュータからの要求は、LAN内にブロードキャストされ、見つからなかった場合には、クラスリクエストエージェントと呼ばれるオブジェクトが、広域ネットワーク上のクラスオブジェクトを検索し、クラスを探す[5]。

### 3.2 他言語インタフェースと外部プロセス

他言語インタフェースとしてOZ++言語はC言語の記述を許している。これはインラインC文と呼ばれ、その中でOZ++のデータにC言語プログラムからアクセスする手段も合わせて提供されている。これにより、開発者は、ファイルI/Oやソケットプログラミングのための API を呼び出すことで、新しい基本的なクラスライブラリを提供することができる。また、他言語で記述された既存のライブラリの利用も可能となる。しかし、セ

キュリティの観点から見ると、インラインC文は危険である。OZ++言語は、ポインタの操作を禁じるなど安全性を考慮しているが、インラインC文を使えばポインタ操作によってエグゼキュータを破壊できるし、直接システムコールを呼び出すこともできるからである。

また、OZ++プログラムは外部プロセスとのインタフェースを持つため、外部プロセスとして他言語プログラムを実行することができる(OZ++システムのGUIはこれを用いて実装されている)。これらの他言語プログラムもクラスの一部として3.1節で述べたクラス配送の対象となるが、これらがファイル入出力などを行う可能性もある。

### 3.3 マルチユーザ環境としてのOZ++のセキュリティ

LAN上のマルチユーザ環境として、OZ++は、次のことがらを満たさなければならない。

1. OZ++で蓄積された重要な情報やプライバシーにかかわる情報に不正なアクセスが発生しないこと
2. サービスの管理者以外の人間が故意あるいは過失によりサービスを停止したりアクセス不能にしたりできないこと
3. 信頼のおけるサーバを偽サーバで置き換えて偽のサービスを行えないこと

これらは、オブジェクトやオブジェクトに関係するファイルをユーザの権限に基づくアクセス制御により保護したり、オブジェクトの身元確認を行うことで対処できる。3.3.1節で、オブジェクトへのアクセスの制限と認証の仕組みを、3.3.2節でファイルへのアクセスの制限の仕組みを述べる。

また、OZ++を利用することで、既存の環境のセキュリティが損なわれることがあってはならない。つまり、OZ++以外で蓄積された情報が流出したり変更されたりしないことが保証されなければならない。3.3.4節で、既存環境の保護について述べる。

#### 3.3.1 オブジェクトへのアクセスの制限と認証

オブジェクトは世界中でユニークな識別子(OID)を持っているので、OZ++のメソッド呼び出しは、LANや広域ネットワークを介しても行われる。そこで、重要な情報やプライバシーに関わる情報の、メソッド呼び出しによる不当なアクセスを禁止するため、オブジェクトごとにアクセス制御を記述できる仕組みを提供する。

身元や権限の確認はオブジェクトのオーナに基づいて行われる。エグゼキュータは、それを作成したユーザをオーナとし、エグゼキュータ上のオブジェクトのオーナはエグゼキュータのオーナであるとする。こうすることにより、アクセス制御に必要な認証プロトコルの実行を、エグゼキュータ間のメソッド呼び出しに限定することができる。

しかし、メソッドの引数や返り値としてオブジェクトが受け渡されることもあるので、オブジェクトのオーナをエグゼキュータのオーナと同一視するだけでは、他人の送り込んだオブジェクトが自分の権限で重要な情報にアクセスできてしまう(図1)。かといって、オブジェクトごとに身

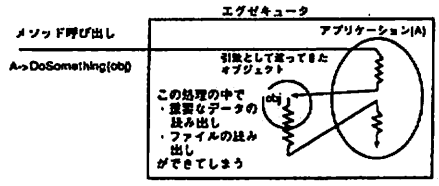


図 1: コピーインされたオブジェクトの危険性

元を管理してアクセス制御を行うのでは、同一エグゼキュータ内でも認証の必要が出て性能上問題がある。そこで、よそから来たオブジェクトや、それらが生成したオブジェクトはuntrustedという属性を持つようにし、その属性を持つオブジェクトが行うアクセスは、オーナuntrustedの権限で行われるようにした。untrusted属性は、インラインC文を用いない限り書き換えることができない。インラインC文の扱いについては、4.3節で述べる。

アクセス制御はサービス側がアクセスリストを保持する方式とし、それを記述する手段はクラスライブラリとして提供することにした。

認証に関しては、通常のLAN環境にはユーザ間で互いに認証を行う仕組みがないため、何らかのトラスティッドサードパーティー方式の認証機構が必要となる。実績の点からKerberosのようなKDCタイプの認証機構[6]を選択することとし、設計を進めている。

### 3.3.2 ファイルアクセスの制限

ファイルアクセスを制限しないと、オブジェクトの永続性の実装に用いられているファイルやアプリケーションが独自にハンドリングしているファイルを読み書きすることにより、オブジェクトへのアクセスを制限する意味が失われる。

そこで、エグゼキュータのオーナを、OSのアクセス制御の対象となるユーザに限り、ファイルに対してOSのアクセス制御が適切に働くように運用する方針を採った。また、untrustedオブジェクトのメソッド内では、一切のファイルの読み書きを禁止することとし、外部プロセスを実行した場合にもそのプロセスのユーザをoznobodyという特別なユーザに設定することにより、エグゼキュータユーザがオーナであるファイルへのアクセス制御を可能とした。

以上、前節と本節ではuntrusted属性を持つオブジェクトに対する制限を述べた。これにより性能を著しく劣化させることなくセキュリティレベルを上げることができるが、アプリケーション構築には制約が課せられることになる。3.3.3節でこの問題への対処法を述べる。

また、インラインC文を用いれば、直接システムコールを発行することにより、ファイルへのアクセスが可能になる。インラインC文の扱いについては、4.3節で述べる。

### 3.3.3 untrusted属性を考慮したアプリケーションの構築

untrustedなオブジェクトが行う処理は、ユーザuntrustedの権限となり、また、ファイルへのアクセスを禁止されるため、そのままでは必要な処理を行えない場合もある。

そのようなケースではアプリケーションを、オブジェクトを受けとる部分と、それを待ち受けていて処理を行う部分のふたつに分けて構築することとなる。送信されるオブジェクトが可能な処理はデータを供給することに限定し、受けとった側が自分の権限で処理を行うようにプログラミングする(図2)。

### 3.3.4 既存環境の保護

マシンには、OSや通常のLANサービス以外の方法でアクセスされることを想定せずに様々な情報が置かれている。OZ++のように、ユーザが特に意識しないうちにプログラムが外部からやって来て実行され得る環境を導入するために、すべてのファイルのアクセス制御を調整し直すのは現実的ではない。そこで、OZ++専用の領域をファイルシステム上に設け、OZ++アプリケーションやOZ++のユーティリティ群は、この領域以外へはアクセス出来ないことにした。

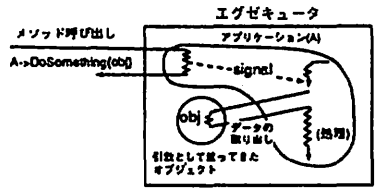


図 2: untrusted属性を考慮したアプリケーションの構築

また、ネットワークアクセスを無制限に許すと既存環境のセキュリティを損なうことになるため、untrustedオブジェクトは既存のネットワークサービスにアクセスできない、という制限を設けた。

## 4 プログラムの配送に関わるセキュリティ

自動配送されたプログラムに基づいて実行するシステムでは、配送の過程でプログラムが不正にすり換えられていないかどうか(保全性)が、セキュリティ上問題となる。また、プログラムを使用するかどうかは、開発元をどの程度信用するかに依存するのは通常と同じである。

また、保全性や身元の保証が得られなくてもプログラムを利用したい場合もある。このような場合でも最低限3.3節で述べたセキュリティを損なわないなど、プログラムの安全な動作を保証する必要がある。

以下では先ず、保全性やプログラムの身元保証の問題にどのように対処し得るかを述べ、その上で安全な動作の保証をいかにして行うかを述べる。

### 4.1 プログラムの保全性

クラスが単一の開発主体によって開発、管理され、配送を経ずに実行される場合にはクラスが改変されることはないと仮定できる。しかし、3.1節で述べた仕組みでは、効率上の理由から、他から配送されてきたクラスのコピーも、要求があれば配送の対象とする運用が可能となってい

る。このとき、悪意のあるクラスオブジェクトの管理者が、クラスのコピーを改変しているかも知れない。LAN内ではクラスの配送が、信用されているクラスオブジェクトからしか起こらないように、認証を用いて慎重に運用することで、対処が可能である。しかし、広域ネットワークを介してクラスが配送されてくる場合には、確実な対処は難しい。

クラス開発者が信頼できる経路でチェックサムを予め配布し、LANで参照可能としておけば、実行する側で計算したチェックサムと照合し、配送の過程で改変されたかどうかをチェックできる。OZ++においては、これを次のように実現できる。

- チェックサムを配布するLAN上のサービスを提供する。これをチェックサムサーバと呼ぶ。
- チェックサムサーバへの(クラスID, チェックサム, レベル, 開発者名, 連絡先)の登録は、LAN上のユーザが3.3.1節で述べた認証を経て行うか、あるいはチェックサムサーバの管理者が行う。
- レベルには、1(テスト中)と2(安定)があり、チェックサムサーバに登録されていないクラスはレベル0とする。
- クラスオブジェクトには、レベル2のクラスのみをサービスするもの、レベル1以上のクラスをサービスするもの、すべてのレベルのクラスをサービスするものの三種類の指定ができる。

チェックサムの計算には、同一値をとるプログラムの偽造が困難なように、一方向ハッシュ関数を用いる。

利用者はエグゼキュータをセットアップする際に、そのエグゼキュータにクラスを供給するクラスオブジェクトの種類を指定することで、配送されてくるクラスのレベルをコントロールできる。

## 4.2 プログラムの信用と責任の所在

プログラムを信用するということは、そのプログラムの開発者を信用することである。そのためには、プログラムが改変されていないということが保証されていなければならない[7]。前節で述べたチェックサムサーバを用いれば、プログラムの保全性が保証できるので、チェックサムサーバの管理者が信用する開発者のプログラムのみを利用可能にできる。

また、開発者名が明らかなのであるから、危険なクラスを流通させようとする意図を持ちづらくすることができるし、万が一事故が発生しても責任の所在を明確にすることができるという効果も期待できる。

## 4.3 安全な動作の保証

チェックサムサービスが充実していないと、責任の所在や保全性が確認できなくてもプログラムを利用したい場合が生じる。その場合、プログラムがOZ++言語で作成されていることすら確かではない。OZ++コンパイラはマシンコードを生成し、エグゼキュータはそれをダイナミックロードして実行するので、そのマシンコードがOZ++コンパイラで開発されたものでない場合、エグゼキュータを停止させてしまうおそれがある。また、C言語のようなポインタ操作が行える言語で開発された場合、untrusted属性を書き換えたりサーバが持つ情報を読み出したりすることが可能となる。これらの問題を防ぐためにはOZ++コンパイラにより自動的に再コンパイルすることが必要となる。再コンパイルの際には、呼び出す側がコンパイル時に指定したインタフェースに基づいて呼び出される側のメソッドの型チェックが行われるので、その動作は保証される。

また、再コンパイルは移入モードという特別なモードで実行されるので、ソースコードにインラインC文が含まれているとコンパイルに失敗し、呼び出し側のアプリケーションに例外が上がる。インラインC文を含むクラスライブラリの配布は、クラス配送とは別経路で行い、通常モードでのコンパイルを経て利用されるものとする。このことにより、インラインC文でセキュリティを脅かす行為を防止できる。

以上の対処により、3.3節で述べたセキュリティは、オブジェクトの交換に従って、たとえ全く素性の知れないプログラムが配送されてきたとしても破られ得ないことが保証できる。

配送の度に再コンパイルが行われるのではプログラムの実行が遅くなるが、認証により信頼できるクラスオブジェクトから配送されたことが確認できたクラスについては再コンパイルを行わないなどのチューニングが可能である。また、現在、再コンパイルの時間短縮のために、中間表現でのチェックおよび実行や両者の併用を検討して

いる。

## 5 OZ++のインターネットを介した運用

厳しいセキュリティ方針を持つ組織は、ファイアウォールと呼ばれる機器の集まりを介して、インターネットと組織内部の保護されるべきネットワークを接続するのが一般的である。そこで、そのような組織がインターネットを介してOZ++を利用する場合、OZ++のメソッド呼び出し、クラス配送などの通信パケットは、ファイアウォール上のリレープログラム（プロクシとも呼ばれる）によって中継される、という形態をとらなければならない。そこで、OZ++システムの一部として、そのようなプロクシを開発する。

### 5.1 OZ++サイト

OZ++の運用単位をOZ++サイトと呼ぶ。OZ++サイトは管理権限が明確な、ひとつの組織によって運営されていなければならない。組織は複数のOZ++サイトを運営できる。

ファイアウォールを介してインターネットと接続する場合、ゲートウェイの存在するネットワークは非武装地帯と呼ばれ、保護される内部ネットワークと区別される。ファイアウォールのない環境ではネットワーク全体が非武装地帯である。OZ++サイトは、非武装地帯上で運用される場合もあるし、内部ネットワークで運用される場合もある。

### 5.2 プロクシによるOZ++サイトの運用

プロクシはゲートウェイマシン上で実行され、他組織のプロクシや、自組織の、ひとつまたは複数のOZ++サイトに対して次のサービスを行う。

1. OZ++サイト外部のエグゼキュータの通信アドレスを教える。
2. 外部のプロクシに、自分が管理するOZ++サイト内のエグゼキュータの通信アドレスを教える。
3. OZ++サイト外部へ（から）のメソッド呼び出しや返り値を中継する。
4. クラスファイル転送パケットを中継する。

1, 2はどちらも3.3.1節に述べたOIDをキーとして行われる。OIDにはOZ++サイトとエグゼキュータを同定するための数値(ID)が含まれている。また、1を実現するために、OZ++サイトデータベースという、OZ++サイトIDとプロクシの通信アドレスの対の表をプロクシに供給するサービスを運用する。図3に、OZ++サイトとプロクシの運用例を示す。非武装地帯での運用の場

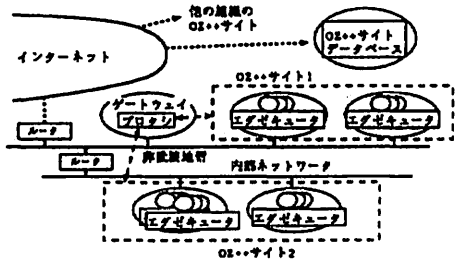


図 3: OZ++サイトとプロクシの運用例

合、プロクシには1, 2のみを行わせ、通信内容の中継を行わせない運用も考えられる。このような運用をオープンな運用と呼ぶ。オープンでない運用はクローズドな運用である。クローズドな運用のとき、2が返すものは、そのプロクシ自身の通信アドレスである。

## 6 OZ++のインターネットセキュリティ

3.3節、4.3節で述べた機構をインターネットを介した運用においてもうまく働くようにすれば、2節で述べた要件は満足される。

また、組織によっては、内部ネットワーク内のマシンに外部からのアクセスがあることを許さないセキュリティ方針を持つ場合もある。そこで、クローズドな運用に、さらに二レベルを設けることとし、6.2節にまとめる。

4.1節、4.2節で述べた機構により、自動配送による実行を、身元が確かで保全性が保証できるプログラムに制限することにより、さらにセキュリティレベルを上げることが可能である。

組織内部での運用においては、サービスを要求するユーザの権限にもとづくアクセス制御や利用するサーバの身元確認においてトラスティドサードパーティ方式の認証に依存した。しかし、インターネットを介した運用を考える場合、広域認証をどうするかが大きな問題である。我々は、OZ++のインターネットを介した利用とその際のセキュリティを検討する上で、広く利用可能な広

域認証システムが存在しないため、現時点では、次のように広域認証自体を用いない方針を採った。

- OZ++サイトの外からのサービス要求は、ユーザのアイデンティティによってではなく、「外来である」という属性のみによってアクセス制御される。これについては次節で述べる。
- チェックサムサーバへの自動登録はOZ++サイトの外からは行えない。これについては7.2節で述べる。

### 6.1 外来性に基づくアクセス制御

引数や返り値でやって来るオブジェクトに関しては、3.3.1節、3.3.2節で述べた機構が適切に働く。しかし、OZ++サイト内部で利用しているサービスをそのまま外部にも利用可能とした場合、メソッド呼び出しにより、認証機構やアクセス制御機構が破られるという脅威にさらされることになる。また、組織内部のOZ++アプリケーションのアクセス制御の連鎖に予期せぬ不備があった場合には、確実に情報の流出やサービスの妨害が発生する。そもそもファイアウォールでの運用自体が、内部ネットワーク内のシステムがセキュアでないことを前提としている[1]のだから、完全なアクセス制御を期待することには無理がある。

そこで、プロクシにより、外部からのメソッド呼び出しや、外部へのメソッド呼び出しに対する返り値のパケットには外来を示す印をつけ、外部起源のオブジェクトやメソッド呼び出しすべてに外来の属性が伝播されるようにする。外来属性を持つメソッド実行は、3.3.1節に述べたアクセス制御において、常にユーザuntrustedの権限となる。さらに、untrusted属性の場合とは異なり、外来属性を持つメソッド呼び出しの連鎖は、ユーザuntrusted以外の権限を決して獲得しない。同様に、外部起源のメソッド呼び出しの連鎖は、決してファイルを読み書きできない。従って、OZ++サイト内部のアクセス制御に穴があったとしても、機密性の高い情報を外来のオブジェクトがアクセスすることはない。

これにより、2節に述べた要件は満たされる。

### 6.2 運用レベルによる保護

組織によっては、内部ネットワーク内のマシンに外部からのアクセスがあることを許さないセキュリティ方針を採っている場合がある。そこで、外部からのメソッド呼び出しを受け入れる運用と、受け入れない運用の二レベルを設けることとし、それぞれ保守的運用、進歩的運用と呼ぶ。保守的運用では、プロクシにより、OZ++サイト間でのメソッド呼び出しの向きを、内側から外側へのみに制限する。このことにより、外部から利用可能なサービスを提供することはできなくなるが、組織の外部からのメソッド呼び出しによる予期しないサービスのアクセスを完全に排除できる。それでもまだ、サービスを利用した場合の返り値としてやってくるオブジェクトが外部の悪意を担っている可能性はあるが、進歩的運用に比べ、その機会は少なくなる。また、使用不能攻撃の機会も少なくなる。

| | | | ネットワーク | |
|---|---|---|---|---|
| | | | 非武装地帯 | 内部ネット |
| 運用 | オープン | | ○ | × |
| | クローズド | 進歩的 | ○ | ○ |
| | | 保守的 | ○ | ○ |

表 1: ネットワークの形態と運用レベル

表1に、ネットワークの保護の形態とOZ++サイトの運用のレベルとをまとめる。内部ネットワークにおいてオープンな運用をするのは不可能であるが、他の組合せは全て可能である。

しかし、オブジェクトを送り合うメールシステムなど、メソッド呼び出しによってオブジェクトが送られて来ることが必須のアプリケーションの実装を考えると、外部からのメソッド呼び出しの受け口が、やはり必要になる。この場合にはこうした受け口専用のOZ++サイトを進歩的に運用し、保守的運用のクライアントとの間でリレー用のオブジェクトを運用する形態を採ることとなる(図4)。

## 7 今後の課題

### 7.1 広域認証の利用

KDCタイプの認証サービスは、ふたつの認証サーバの管理者が、予めひとつの鍵を共通に知っておくことで、ドメインを越えて運用できること

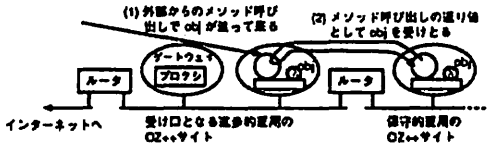


図 4: 保守的運用でのオブジェクトの取り込み

が知られている[6]。OZ++でも、当面、この方法（仮に会話鍵共有方式とよぶ）の採用を検討している。しかしこの方法はスケーラビリティに欠けるので[8]、公開鍵方式で認証プロトコルを拡張する方法が提案されている[9]。

### 7.2 チェックサムサービスの広域での運用

LANのチェックサムサーバの内容を、その管理者が人間的なやりとりを通じて充実させてゆくのでは限界がある。そこで、開発者にチェックサムを登録させる認定機関が設立されると仮定する。各組織のチェックサムサーバの管理者がその機関から郵送などでチェックサムリストを入手し、自分の組織のチェックサムサービスにその内容を反映させれば、責任の所在が明確で、保全性を確認できるプログラムをインターネットで広く共用することが可能になる。

このとき認定機関がチェックサムリストサーバを運用し、各組織がそれを通信によりダウンロードできれば便利である。用途をこの、チェックサムリストサーバのダウンロードに限れば、前節で述べた会話鍵共有方式の認証で、このようなサービスを運用することができるであろう。

## 8 まとめ

プログラムの自動配送に基づいてアプリケーションを実行する環境を広域ネットワークを介して利用しても、既存環境のセキュリティを損なわないし、重要な情報の流出やサービスの破壊、資源の不正利用が生じないことをOZ++を例に示した。また、プログラムの身元保証を用いて信用に裏打ちされたプログラムの利用が可能となること、運用の様々なオプションにより、それほど利便性を損なわずに組織のセキュリティ方針をカバーする方法があることを述べた。

OZ++は、第一版[10]を既にWWWで公開しており、今後、本稿で示した検討内容を踏まえ、LAN内でのマルチユーザ対応やインターネット環境での利用に向けて拡張を行う予定である。

## 謝辞

本研究を通じて熱心な討論をいただいている当プロジェクトのメンバー諸氏に感謝する。

本研究は、情報処理振興事業協会(IPA)が実施している「開放型基盤ソフトウェア研究開発評価事業」の一環として行われたものである。

## 参考文献

[1] Treese, W. and Wolman, A.: "X through the firewall, and other application relays", In Proceedings of USENIX Conference, pp. 87-99, Jun. 1993.

[2] Cheswick, W.R. and Bellovin, S.M.: "Firewalls and Internet Security : Repelling the Wily Hacker", Addison-Wesley, 1995.

[3] 西岡他: 「オブジェクト指向分散環境OZ++の言語の基本設計」、情報処理学会第46回全国大会、Mar.1993

[4] 浜崎他: 「オブジェクト指向分散環境OZ++の実行機構の設計」、情報処理学会第48回全国大会、Mar.1994.

[5] 西岡他: 「オブジェクト指向分散環境OZ++のクラス配送機構」、情報処理学会第50回全国大会、Mar.1995.

[6] Steiner, J., Neuman, C. and Schiller, J.I.: "Kerberos: An Authentication Service for Open Network Systems", In Proceedings of Winter USENIX Conference, Dallas, 1988.

[7] Rubin, A.D.: "Trusted Distribution of Software Over the Internet", In Proceedings of the Symposium on Network and Distributed System Security, IEEE, San Diego, pp. 47-53, Feb. 1995.

[8] Bellovin, S.M. and Merritt, M.: "Limitations of the Kerberos Authentication System", In Proceedings of USENIX Conference, Dallas, pp. 253-267, 1991.

[9] McMahon, P.V.: "SESAME V2 Public Key Authorization Extentions to Kerberos", In Proceedings of the Symposium on Network and Distributed System Security, IEEE, San Diego, pp.114-131, Feb. 1995.

[10] 西岡他: 「オブジェクト指向分散環境OZ++システム第一版の実現」、SWoPP95、Aug. 1995.