マルチメディア通信と分散処理　101－3
コンピュータセキュリティ　12－3
（2001．2．20）

共生・寄生エージェントモデルにおける認証メカニズム

溝入 優一† 高田 眞吾† 飯島 正† 土居 範久†
{ mizoiri, michigan }@doi.cs.keio.ac.jp,
iijima@ae.keio.ac.jp, doi@keio.ac.jp
† 慶應義塾大学大学院理工学研究科
神奈川県 横浜市 港北区 日吉 3-14-1

あらまし　急速な情報ネットワークの発展に伴い，情報の氾濫やシステムの複雑化が進んでおり，これらに対応できる技術としてソフトウェアエージェントが注目されている．ソフトウェアエージェントとは，人間になり代わって自律的に動作するソフトウェアである．エージェントには，処理の実行中に起こりうる環境の変化に適応できることが期待されるため，エージェントが動的に必要な機能や権限を拡張することのできる共生・寄生エージェントモデルが提案されている．このモデルでは，機能や権限をもエージェントとみなし，エージェント間の共生・寄生によってエージェントの動的な拡張を可能にしている．本研究では，このモデルにおいて適切な共生・寄生関係を築くために必要な認証メカニズムを提案する．
キーワード　共生・寄生エージェントモデル，移動エージェント，認証

Authentication for Symbiotic/Parasitic Agent Model

MIZOIRI, Yuichi† TAKADA, Shingo† IIJIMA, Tadashi† DOI, Norihisa†
† Graduate School of Science and Technology Keio University
3-14-1, Hiyoshi, Kohoku-ku, Yokohama, Kanagawa, JAPAN

Abstract　The growth of the information network has led to much research on its efficient use, one of them being agent technology. A software agent is a software that autonomously behaves on behalf of humans. Since agents need to be able to dynamically adapt to changes in the environment, they need to be able to extend their functions and authorities dynamically. The Symbiotic/Parasitic Agent Model is one model that realizes this through agents having symbiotic/parasitic relations with each other. There is, however, a danger that an agent based on this model may attack other agents and/or resources. This paper proposes an authentication mechanism that considers agents' authorities to protect agents and resources from other agents.
Keywords　Symbiotic/Parasitic Agent Model, Mobile Agent, Authentication

1 はじめに

ソフトウェアエージェントとは，人間になり代わって自律的に動作するソフトウェアである．エージェントには，処理の実行中に起こりうるシステムや資源などの環境の変化に適応することが期待される．特に，ネットワーク上を自律的に移動して，移動先において，送り主の代わりにタスクを処理する移動エージェントには，環境に適応することが強く期待される．

共生・寄生エージェントモデル [2][3] とは，環境の変化に適応するために，エージェントが動的に必要な機能や権限を拡張することのできるモデルである．現時点において，このモデルに認証機構は備わっていないが，外部の攻撃からホストやエージェントを守るためには，適切な共生・寄生の関係を構築し権限を管理する認証機構が不可欠である．そこで，共生・寄生モデルにおける認証メカニズムを提案する．

2 共生・寄生エージェントモデル

共生・寄生エージェントモデル (S/PAM : Symbiotic/Parasitic Agent Model)[2][3] は，機能や権限をもエージェントとみなし，エージェント間の共生・寄生によってエージェントの動的な拡張を可能にしている．環境への適応は，エージェントが，必要な機能を備えたエージェントを自らの内部に寄生させることによって実現する．さらに，このモデルでは，寄生操作を再帰的に繰り返すことも可能であるとしているため，エージェントは入れ子構造をとることができる．

2.1 共生・寄生関係の構築

図 1に，このモデルにおいてエージェント間の関係を操作する基礎的な動作を示す．ここで，**寄生**とは，エージェント (寄生エージェント) が他のエージェント (宿主エージェント) の内部に寄生する動作であり，**脱出**とは，寄生エージェントが宿主エージェントから脱出する動作である．これらの動作以外にも，エージェントが自分の内部に他のエージェントを寄生させる**獲得・吸収**や，宿主エージェントが寄生エージェントを排出する**排出**，エージェントを動的に結合して，一つの「個体」としてのアイデンティティをもったエージェントを構成する**合成・統合**，エージェントが自分の能力・権限の一部を新

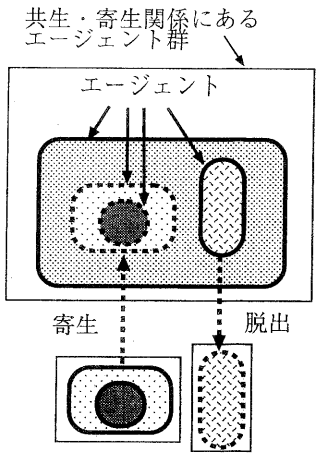

図 1: 共生・寄生エージェントモデルの概念図

しいエージェントとして分割・複写する**分割・委譲**がある．

2.2 共生・寄生モデルでのセキュリティ

悪意のあるエージェントからエージェントや資源を守るために，適切な共生・寄生の関係を築くことのできる認証メカニズムが必要であるが，現時点では備わっていない．また，このモデルは，機能をもエージェントとみなしているため，寄生しているエージェントが自由に外部と通信できると，秘密情報が外部に漏洩する可能性がある．本研究は，これらの問題を解決するものである．

3 移動エージェントのセキュリティ技術

移動エージェントの実用化にはセキュリティの確保が不可欠であり，現在，多くの研究が行われている．本節では，本研究で参考としたセキュリティ技術について述べる．

3.1 認証とアクセスコントロール

認証とは，移動エージェントを実行するシステムが，エージェントに対するデジタル署名を検証し，署名者を特定することである．この時，署名の検証により，エージェントが移動中に改ざんされていないことも確認できる．アクセスコントロールとは，認証の結果からエージェントに与える権限を決定し，エージェントをこの権限の範囲内で実行させることである．

3.1.1 Javaセキュリティアーキテクチャ

Sun Microsystems社のJavaセキュリティアーキテクチャには，ネットワークを介して受け取るコード（アプレット）からホストを守るために，sandboxモデルと呼ばれる認証とアクセスコントロール機能が備わっている[4]．JDK 1.2 (Java Development Kit 1.2) におけるsandboxモデルでは，ローカルコードとリモートコードのどちらの場合でも，コードに割り当てられる権限は，コードを読み込むクラスローダとセキュリティポリシによって決定される．セキュリティポリシとは，Java Virtual Machineを実行する主体が，どのコードに対してどの権限を与えるかを記述したものである．セキュリティポリシには，コードをロードする位置 (URL) とコードへの署名者の組に対して許可する権限を記述している．そして，コードが権限チェックの必要な処理を行う場合，セキュリティマネージャ（アクセスコントローラ）が呼び出され権限のチェックが行われる．

ここで，注意しておきたいことは，Javaのセキュリティモデルが対象としているものは，コードであってオブジェクトではない．つまり，エージェントのように内部状態を保有する主体に対して権限を決定する場合には，この方法では不十分である．

3.2 移動エージェントの認証

Berkovitsらは，エージェントが信頼できるプレースを移動することによって，ホストからエージェントを守る方法を提案している[1]．彼らは，移動エージェントシステムを構成する主体として，次の5つを挙げている．

1. **プログラマ** エージェントの元となるプログラムの作成者
2. **署名付きプログラム** プログラムにプログラマの署名などの情報を付加したもの
3. **センダ** エージェントの生成者
4. **署名付きエージェント** エージェントにセンダの署名などの情報を付加したもの
5. **プレース** ホスト上の実行環境

プログラマは，自分の作成したプログラムを元に，署名付きプログラムを作成する．署名付きプログラムには，そのプログラムからエージェントを生成できるセンダのリストであるSender Permissioon List (SPL) がプログラムに連結されて署名されている．次に，センダが署名付きプログラムを元にして署名付きエージェントを生成する．この時，エージェントの信頼するプレースのリストであるPlace Permission List (PPL) をエージェントに連結して署名を行う．そして，エージェントは，SPLやPPLで指定された関係が満たされている場合のみ実行される．エージェントを攻撃から守るためには，このように作成者が信頼情報をプログラムやエージェントに埋め込む方法が有効であると考え，本研究ではこの考えを利用する．

4 共生・寄生モデルにおける認証メカニズム (定義と課題)

本研究における定義と課題について述べる．

4.1 システムの中心となる主体

エージェントシステムの中心となる主体を次のように定義する．

1. **プログラマ** エージェントの元となるプログラムの作成者
2. **コード** プログラムに，プログラマの署名などの情報を付加したもの
3. **センダ** 移動エージェントの生成者
4. **移動エージェント** エージェントに，センダの署名などの情報を付加したもの
5. **共生エージェント** 共生・寄生関係にある移動エージェントの集合に，署名などの情報を付加したもの
6. **ホスト** エージェントを実行するホスト

共生エージェントとは，共生・寄生関係にある移動エージェント群がセンダ-ホスト間やホスト-ホスト間を移動する時の形態である．これは，**移動エージェント群**と，共生・寄生エージェントの**送り主名**と，**受け取り主名**を連結したものに，送り主の署名を加えたものである．

共生・寄生モデルでは，実行環境であるプレースもエージェントとみなすため，実際にエージェントを実行するのはマシン上でエージェントシステムを提供する主体である．この主体をホストと呼ぶ．

プログラマとセンダとホストは，それぞれ公開鍵暗号方式の公開鍵と秘密鍵の組を保有するが，コードと移動エージェントは，内部に自分の鍵を保有しない．なぜなら，信頼度の異なるホストを移動する

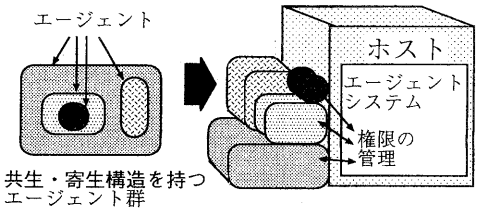

図 2: 権限の管理

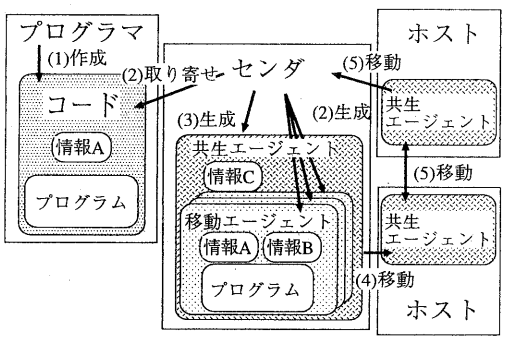

図 3: エージェントシステムの概要図

移動エージェントにとって，署名を行う鍵を内部に保持しネットワークを移動することは危険であるからである．

4.2 権限の管理

権限の管理は，図 2のように表現できる．ホストはすべてのエージェントに対して直接システムを提供しており，権限は共生・寄生関係にある一番外側のエージェントのみに割り当てられるではなく，各々のエージェントに対して割り当てられる．そして，エージェントは，割り当てられた権限の範囲内で処理を行う．

4.3 エージェントのライフサイクル

図 3を用いてエージェントのライフサイクルを述べる．

(1) プログラマは，プログラムを作成し，それに署名などの情報 (情報 A) を付加したコードを作成する．

(2) センダは，コードを取り寄せ，これを元に署名などの情報 (情報 B) を付加し移動エージェントを生成する．

(3) センダは，1つ以上の移動エージェントに，署名などの情報 (情報 C) を付加した共生エージェントを生成する．

(4) センダは，ホストに共生エージェントを送り，ホストはエージェントを実行する．

(5) 共生エージェントは，必要があればホストやセンダに移動する．

4.4 目的

本研究で提案する認証メカニズムは，特に次の目的を持つ．

- 適切な共生・寄生関係を構築・保持することができる．
- エージェントが，自分に寄生させているエージェントに対して，外部との通信などの権限を制限できる．

5 共生・寄生モデルのための認証メカニズム (ポリシと寄生のモード)

ホストが，移動エージェントの内部にあるセキュリティポリシを用いて，適切な共生・寄生関係を構築・保持する方法と，寄生行動に2つのモードを導入して，エージェントが保有できる権限を制限する方法を提案する．本節では，まず，この2つの方法について述べ，次に，これらの方法に基づくエージェントの検証と権限の割り当ての手順を述べる．

5.1 セキュリティポリシ

本システムでは，3つのセキュリティポリシを利用して，エージェントの関係を構築し権限を決定する．エージェントが実行されるのは，プログラマが自分の作成したコードの信頼情報を記述したプログラマセキュリティポリシと，センダが移動エージェントの信頼情報を記述したセンダセキュリティポリシが満たされている場合のみである．また，ホストがホストセキュリティポリシで許可している範囲内でエージェントは実行される．

5.1.1 プログラマセキュリティポリシ

プログラマセキュリティポリシは，プログラムに連結され，プログラマによって署名される．このセキュリティポリシは，付随するプログラムを元に生成する移動エージェントに関係する次に挙げるような信頼情報のリストの集合である．

- Host Permission List (HPL_p)

  移動エージェントが移動してもよいホストのリストである．

- Sender Permission List (SPL)

  移動エージェントを生成することのできるセンダのリストである．

- Agent Permission List (APL_p)

  移動エージェントに対して寄生させることを許可する移動エージェントのリストである $getAPL_p$ と，生成された移動エージェントが寄生することを許可するリストである $enterAPL_p$ から構成される．

5.1.2 センダセキュリティポリシ

センダセキュリティポリシは，エージェントに連結され，センダによって署名される．このセキュリティポリシは，次に挙げるリストの集合である．

- Host Permission List (HPL_s)

  HPL は，プログラマセキュリティポリシの HPL_p と同様に移動エージェントの移動を許可するホストのリストである．ここで，移動エージェントが，プログラマの許可していないホストに移動することは不適切であるため，HPL_s は，HPL_p の部分集合でなければならない．

- Agent Permission List (APL_s)

  APL_s も，プログラマセキュリティポリシの APL_p と同様のものであり，HPL と同じ理由で，APL_s は，APL_p の部分集合でなければならない．

- Code Permission List (CPL)

  CPL も APL と同様に共生・寄生関係を構築するための信頼情報を記述したものであり，APL と同様に getCPL と enterCPL がある．APL が寄生関係を築くエージェントを制限しているのに対し，CPL は寄生関係を築くエージェントの元となるコードを制限するものである．

5.1.3 ホストセキュリティポリシ

プログラマとセンダの組に対して，ホストが管理するリソースの権限のうちエージェントに割り当てる権限を記述したものである．

5.1.4 リストの有効範囲

エージェントは入れ子構造をとることができるため，APL と CPL には有効範囲 (有効な深さ) の設定が必要である．getAPL と getCPL に関しては，寄生するエージェントの内部に寄生しているエージェントを含むすべてのエージェントが，リストに記述されていなければならないとする．なぜなら，寄生させるエージェントが寄生するエージェントを利用するということから，利用する側は信頼できるエージェントしか利用しないべきであるためである．これに対して，enterAPL と enterCPL に関しては，寄生させるエージェントを寄生するエージェントが十分に信頼している場合にのみ関係を築くとし，直接関係するエージェントのみがリストに記述されていればよいとする．

5.2 寄生のモード

エージェントは，寄生させているエージェントに対して通信や脱出の権限を制限しなければならない場合がある．つまり，寄生しているエージェントに，秘密情報を渡す場合である．このような時，寄生しているエージェントに悪意がなくても，通信や脱出の権限を保有していると情報が外部に洩れる可能性がある．通常，通信などの権限は，ホストセキュリティポリシによってエージェントに割り当てられるが，ホストとエージェントの関係のみではなく，共生・寄生の関係によってもエージェントに与える権限を決定する必要がある．

本研究では，通信や脱出に制限を加えるモードである部品エージェントモードと，制限を加えないモードである共生モードに寄生行動を分類する．図 4に，寄生のモードの例を示す．ここで，アルファベットは，エージェント名を示す．図 4では，Agent B や Agent C のように部品エージェントモードで寄生している場合には，脱出の権限などを保有しない．寄生している．それに対して，共生モードで寄生している Agent E や Agent F は脱出する権限などを保有しており，他のエージェントに寄生することができる．もちろん，この寄生行動が適切なものであるかを検証する必要があり，それぞれの寄生のモード別に，セキュリティポリシ内の APL と CPL を記述する必要がある．

この時，部品エージェントモードで寄生するエージェントは，内部に共生モードのエージェントを寄生させていてはならない．なぜなら，その内部に寄生している共生モードのエージェントを使って，外部と通信を行う可能性があるためである．

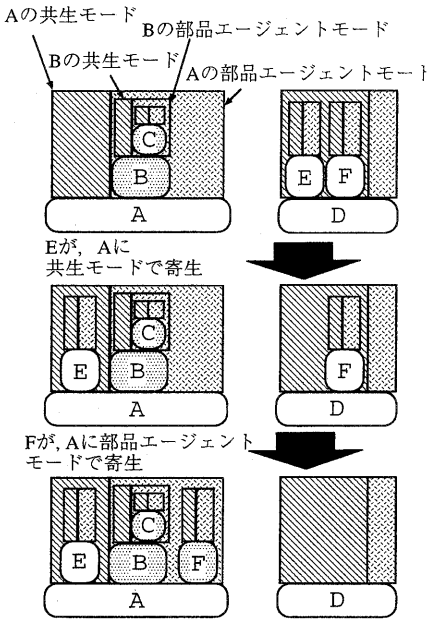

図 4: 寄生のモード

5.3 検証と権限の割り当ての手順

検証と権限の割り当てが必要となるのは，エージェントが移動する時である．エージェントの移動は，センダ - ホスト間やホスト - ホスト間の移動と，同じホスト内で共生・寄生関係を変化させるための移動に分かれる．前者は，共生エージェントとして移動する．この時，共生エージェントの受け取り主は，まず，共生エージェント内の移動エージェントの検証を行う．検証の結果，正当なものであれば，希望するホスト内の移動エージェントに対して寄生できるかの検証を行う．これに対して，ホスト内の移動の検証は，希望するホスト内の移動エージェントに対して寄生できるかの検証のみでよい．なぜなら，ホスト内のエージェントは，一度検証してあるものとし，エージェントの実行によって変化しない部分は，書き換えられないように設定しておくためである．次に，ホストに到着した共生エージェントに対する検証と権限の割り当ての手順を示す．

1. 共生エージェント内の全ての署名を検証する．
2. 共生エージェント内の移動エージェントが保有するすべてのプログラマセキュリティポリシとセンダセキュリティポリシに記述されている信頼関係が成り立っていることを検証する（共生エージェントの送り主も移動エージェントを実行できることを検証する．）．
3. ホストセキュリティポリシを用いて，共生エージェント内の移動エージェントに与える権限を決定する．
4. 共生エージェント内に部品エージェントモードの寄生関係があれば，対象となるエージェントの権限のうち許可されないものを削除する．
5. 共生エージェント内の移動エージェント群が希望するエージェントに寄生しても，関係するプログラマセキュリティポリシとセンダセキュリティポリシが満たされることを検証する．
6. 上記の寄生が部品エージェントモードであれば，寄生するエージェントの権限のうち許可されないものを削除する．
7. 割り当てられた権限を用いてエージェントの権限チェックを行う．

この方法を用いると，共生・寄生関係の適切性のチェックと，共生・寄生関係によって動的に変化する移動エージェントの権限の制御ができる．

6 まとめ

共生・寄生エージェントモデルにおいて，適切な関係を構築し，エージェント間の関係を考慮した権限の割り当てを行うメカニズムを提案した．

参考文献

[1] Berkovits, S., Guttman, J.D., and Swarup, V., “Authentication for Mobile Agents”, *Mobile Agents and Security*, LNCS, vol. 1419, 114-136 (1998)

[2] 飯島 正, 山本 喜一, 土居 範久, “移動エージェントのための共生・寄生モデル”, 第 55 回情報処理学会全国大会, 論文集第 3 分冊 749-750 (1997).

[3] 飯島 正, 山本 喜一, 土居 範久, “拡張可能エージェントを利用する時に渡す情報が漏洩する可能性がある．なエージェントのための共生・寄生モデル”, 電子情報通信学会, 知能ソフトウェア工学研究会 KBSE98-33 (1998).

[4] Sun Microsystems. Trail: Security in Java 2 SDK 1.2. Web page available at http://java.sun.com/docs/books/tutorial/security1.2/index.html.