情報処理学会論文誌 Vol. 50 No. 10 2484–2499 (Oct. 2009)

効率的なセキュリティ要求分析手法の提案

大久保 隆夫[†1,†2] 田中 英彦[†2]

本論文では，開発者全体にセキュリティ知識が浸透していない状況でも，効率的にセキュリティの要求策定を行うことができるセキュリティ要求分析手法を新規に提案する．提案手法の主な特徴は次の 2 点である．1 点目は，セキュリティ知識を持つ者と，ソフトウェアのドメイン知識を持つ者とが分離していることを前提にした，アスペクト指向のセキュリティ要求策定プロセス（AOSRE）である，2 点目は，要求分析の中でも負担の重い作業になっている脅威，対策案の抽出手法に対して，既存のミスユースケース図を新規に拡張し，保護資産やアーキテクチャの概念を追加することで分析者の作業の効率化および結果の共有を容易にした，資産ベースのミスユースケース手法（AsseMis）である．筆者らは提案した手法を実際のアプリケーション開発に適用し，効率化，有効性について評価を行った．

A Proposal of an Efficient Security Requirements Analysis Method

Takao Okubo[†1,†2] and Hidehiko Tanaka[†2]

In this paper, we propose a new security requirements analysis method which enables efficient requirement elicitation even with limited security expertise. The proposed method contains two main features. One is a new aspect-oriented security requirements elicitation process (AOSRE) which is based on the assumption that the security expertise and the domain knowledge are isolated. The other is a new asset based misuse case approach (AsseMis) which extends a misuse case approach to improve efficiency of threat identification and understandability of results. We have applied our new approach to several application development projects evaluated the efficiency.

†1 株式会社富士通研究所
Fujitsu Laboratories Limited
†2 情報セキュリティ大学院大学
Institute of Information Security

1. はじめに

近年，IT 技術の進歩により様々なサービスがインターネットを経由して提供されるようになった．その一方，同じくインターネットを介した攻撃の脅威も高まっている．情報処理推進機構（IPA）が受け付けた日本国内の脆弱性届出件数は，2007 年度だけで 572 件，2004 年度からの累計では 1,749 件に達している[1)]．また 2008 年 4 月には，実際に大規模な SQL インジェクション攻撃により 50 万以上の Web ページが改竄の被害にあっている[2)]．問題の本質的な解決のためには，ソフトウェア開発工程の早期段階からセキュリティを意識し，脅威への対策を意識した開発が必要になる．筆者らが行っている Web アプリケーションプログラムのシステム・インテグレーション（SI）におけるセキュリティ構築支援の経験によれば，現実のソフトウェア開発においてセキュリティの導入が成功していない主な要因としては，次の 2 点があげられる．

- ステークホルダ[⋆1]のセキュリティ知識不足
- 既存のソフトウェア開発手法とセキュリティ開発手法の不整合

セキュリティ要求分析手法としてはすでに提案されている手法がいくつかあるが，上記問題点の双方に十分な解決を与えるものはなかった．そこで本論文では，上記の問題を解決することを目的とした，開発早期からのセキュリティ構築手法を新規に提案する．提案手法は，以下の 2 点を主な特徴とする．

- セキュリティ有識者とドメイン知識を持つ開発者が別個に存在することを前提にした，分業化されたアスペクト指向セキュリティ要求策定プロセス（AOSRE: Aspect-Oriented Security Requirements Elicitation）
- ミスユースケース[3)] を独自に拡張することにより，脅威抽出と対策の十分性の確認を容易にする分析手法（AsseMis: Asset based Misuse case approach）

本論文では，まず 2 章で研究の背景となる現状の問題と解決方針について詳細に述べる．次に 3 章で，セキュリティの要求分析手法について，要求工学的手法，およびセキュリティに特化した手法の双方の特徴と問題点について述べる．4 章では，本論文で提案するミスユースケースを拡張した要求分析手法について説明する．5 章では提案手法の評価を行い，6 章で結論を述べる．

⋆1 顧客，エンドユーザ，開発担当者，プロジェクト管理者などを含む，対象システムに関わるすべての利害関係者

2. 背　　景

本研究の目的は，セキュリティ知識を持つ人的資源が限られている状況で，効率的，効果的にセキュリティ要求を策定するための手法を提示することである．その背景には，前章で述べたようにステークホルダの知識不足，および既存の開発手法との不整合という 2 つの問題がある．本章では，本研究の背景となっている上記 2 つの問題点について詳細に検討，考察を行う．

- ステークホルダのセキュリティ知識不足
  2006 年から筆者らが Web アプリケーション SI のセキュリティ構築支援として行っている活動は，文献 4) や独自に収集した知識に基づき，プログラムの設計，実装においてセキュリティ上の問題点がないかを診断し，助言を与えるものである．診断対象の開発プロジェクトは現在までにのべ 40，開発に携わる人員は数千人規模に及ぶ．過去の支援活動において指摘した脆弱性は，クロスサイト・スクリプティング（XSS[5]）や SQL インジェクション[6] など，比較的よく知られているものが多かった．ソフトウェアの開発現場（通常，複数のソフトウェア開発会社が担当している）においては，主な脆弱性についてある程度知識はあるものの，その必要性，実装のノウハウ，検証方法に関する知識不足が原因で，開発に問題が発生することがしばしばあることを，筆者らは上記活動の経験から得ている．1 章で述べたように，届け出られている脆弱性のうち，SQL インジェクション，XSS という比較的報道される機会の多い脆弱性が大部分を占めることを考慮すると，多くの開発現場において，このような脆弱性の脅威や対策に対する知識不足が問題になっていることが推察される．
  また，開発対象のソフトウェアのセキュリティ要求が不明確な場合が多いことも問題としてあげられる．筆者らの構築支援活動において，セキュリティ診断で脆弱性を指摘したものの「顧客要求にない」という理由で対処が見送られてしまった場合があった．脆弱性の修正にはコストや人員，期間を要するが，元々開発要求にないセキュリティ問題の対処のための新たな予算投入，期間延長について顧客の了承が得られなかったためである．このようにセキュリティ要求が欠落している要因は，顧客や要求の提案分析を行う担当者側のセキュリティ知識不足にあるといえる．
- 既存のソフトウェア開発手法とセキュリティ開発手法の不整合
  知識不足以外の要因としては，セキュリティの開発手法がソフトウェア工学を基にした従来の開発手法とは異なっているという点があげられる．その相違は分析，設計，実装，テストのすべての工程に存在する．たとえば，セキュリティの要求分析では，システムが扱う資産に対する脅威を想定し対策方針を決定する脅威分析を行う必要がある．しかし，この脅威分析は，従来の機能要求を中心とした手法では，想定されない利用（たとえば攻撃者）や想定されない振舞いへの観点が欠落しているため，実現が困難である．UML[7] のユースケース図は，想定される利用者が，機能を「想定された範囲で」利用することを記述したものである．ユースケース図には，「想定されない利用者（すなわち，悪意ある攻撃者）」による，「想定されない利用（攻撃）」は記述されていない．これらの要素を想定し，記述するためには別のセキュリティ的手法が必要である．

セキュリティ知識不足の問題については，すべてのステークホルダに対するセキュリティ教育が長期的には正しくかつ本質的な方策である．しかし，知識を全体に浸透させるまでには長い期間が必要であることが想定される．全体の中でセキュリティ有識者が少数であるという現状をふまえた対策は別途，必要になることが想定される．したがって，少数のセキュリティ有識者と他のステークホルダの存在を前提とした，高品質のセキュリティを維持する効率的な開発手法が有効であると考えられる．仮に，要求分析段階において現状の体制（少人数のセキュリティ有識者）でも効率的な要求定義によりセキュリティリスクを明確化し，ステークホルダ間でセキュリティ要求について事前に合意を得ることができれば，設計以降の段階において発覚したセキュリティ問題への対処を「顧客要件」により拒否されることは回避できる可能性が高まる．本論文では上述のような効果を期待し，特に対象とする開発工程を要求分析段階に絞り，効率的かつ高品質なセキュリティ要求分析手法を提案する．

3. 既 存 研 究

本章では，セキュリティ要求分析における既存研究とその問題点について論じる．

セキュリティの要求分析に対する既存研究，技術は，大別して既存のソフトウェア要求分析手法の延長，応用であるものと，セキュリティに特化した分析手法との 2 種類がある．前者に属するものとして，ゴール指向分析手法，UML の拡張，問題フレームを用いた手法[8),9)]が，後者に属するものとして，SDL[10)]，コモンクライテリア（CC）[11)]，SQUARE[12)]，脅威モデリング[13),14)] があげられる．

3.1 ゴール指向分析

ゴール指向分析手法には，KAOS[15)]，i*フレームワーク[16)]，Secure Tropos[17),18)] などがある．ゴール指向分析では，最初に最上位のゴールを決定し，次にそのゴールをサブゴールに分割していくことで，具体的な要求を策定していく．例として KAOS をあげる．KAOS

の中では，セキュリティはゴールの一種（非機能ゴール）として表現される．この手法は，セキュリティとして満たすべきゴールが明確になっていなければ，まずそのゴールを識別する作業が必要になる．前章で述べたように，現実の SI 開発ではセキュリティゴールが明確でない場合が多い．

このほか，KAOS では悪意ある者のゴールとして，反ゴールを想定し，反ゴールの詳細化を行う手法[19] と，各サブゴールに対する障害要因を想定し，詳細分析していく手法がある．これは，従来の機能要求のみの分析にはない，セキュリティ的観点を導入しようとする試みである．しかし，反ゴールや障害分析にも問題点がある．1 点目は，反ゴールや障害を分解していく際に，セキュリティ知識が要求される点である．2 点目は，反ゴールや障害分析はアーキテクチャが詳細であるほど有効な手法だが，要求分析段階ではシステムアーキテクチャが詳細化がされていない場合が多く，反ゴールや障害の可能性の判断は困難になることが想定される点である．例として，バッファオーバフローをあげる．バッファオーバフローの実現可能性は，実装言語や用いる関数，ソフトウェアの構成，外部からの入力などに依存する．したがって，バッファオーバフローの構造に関する知識やシステムの詳細についての情報がない状況では，脅威の識別や可能性の評価は困難になる．

3.2　UML の拡張

UML を拡張する手法としては，UMLsec[20]，ミスユースケース[3] などがある．

UMLsec　UMLsec は，UML の拡張記法を利用し，セキュリティ要素を UML 上の拡張に反映させることにより，モデルベースの開発を行うための手法である．UMLsec は要求分析よりも，設計段階向けの手法であり，セキュリティを記述する対象は，クラス図やアクティビティ図など設計段階で作成される図が中心である．セキュリティ要求は前提として扱われ，UMLsec 自身が要求策定を行う手段を提供するわけではない．

ミスユースケース　Sindre と Opdahl は，UML のユースケース図を拡張したミスユースケース図を提案している[3]．ミスユースケース図の例を**図 1** に示す．図中で，アクタを黒く塗りつぶしたものが，通常のアクタではなく，意図しない振舞いをする「ミスアクタ」を表し，意図しない振舞いそのものは，ユースケースを黒く塗りつぶした「ミスユースケース」として表現される．図 1 の例では，ネットワーク上で盗聴を行う「攻撃者」および，漏洩という「脅威」がそれぞれ，ミスアクタ，ミスユースケースとして図上に表現される．ミスユースケース図では，脅威に対する対策を通常のユースケースとして表記する．

ミスユースケースには次にあげる利点があると考えられる．

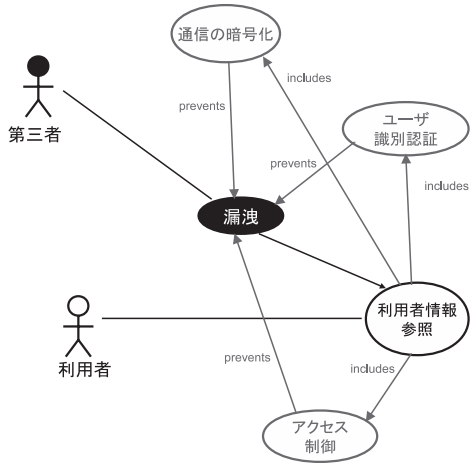

図 1　ミスユースケース図の例
Fig. 1　An example of misuse case diagram.

- 図 1 に示すように，元のユースケースに潜在する脅威と対策を 1 つの図で表現できるため，分析結果がすべてのステークホルダにとって理解しやすい．
- 脅威に対する対策の網羅性の確認が容易である．

このようにミスユースケースは，結果の視認性や理解のしやすさでは優れている．しかし，図上での脅威，対策を抽出する過程は明確に示されてはおらず，攻撃者（ミスアクタ）やミスユースケース（脅威）の導出には，セキュリティ知識を必要とする，セキュリティ有識者にとっても，ミスユースケースが脅威抽出の道具として有効とはいえない．CC や脅威モデリング手法にあるように，脅威分析ではまずセキュリティ上保護すべき資産（改竄，漏洩防止を必要とするデータや，機能妨害を防止する機能．以降単に「保護資産」と呼ぶ）を抽出し，次にその保護資産に対する脅威を想定する手法が一般的に行われている．しかし，ミスユースケース図（ないし，その元になるユースケース）には，ユースケース以外のデータや保護資産の概念が表現されていないため，これらの保護資産に対する脅威は抽出しにくくなっている．

また，ユースケース図は機能の概念を表現する図であるため，アーキテクチャの概念がない．そのため，ミスユースケース図のみでは，脅威の具体的手段である攻撃を想定することが困難になっている．

ミスユースアクティビティ[21),22)]　ミスユースアクティビティは，ユースケースではなく，UML のアクティビティ図上に脅威を拡張する表現を加えたものである．ユースケースを分解しているため，ユースケースレベルよりもより詳細なインタラクションや手順のタイミングに沿った脅威の抽出が可能になっている．しかし，それはユースケースをアクティビティレベルに分解することが必要となる．しかし一般にはアクティビティの詳細化は要求分析ではなくその後の設計段階に属する作業とされているため，要求策定の段階では，各アクティビティの詳細化は期待できない．

3.3　問題フレームを用いた手法

問題フレームは，機械（ソフトウェア）と問題領域，要求の関係を記述することで，ソフトウェアの要求を定義する手法である[23)]．ソフトウェアの要求定義に問題フレームを用いる手法は，元々の問題領域が，エレベータのような典型的な振舞いをする対象には適しているが，1 回だけの開発，納品の多い SI のような分野では，問題フレームに記述されるレベルでは再利用がされにくい．より問題を詳細に分解すれば，再利用される可能性は高くなるかもしれないが，それは，そこまでの詳細な構造になることが，要求分析段階で明らかになっている場合に限られる．また，Haley らは，問題フレームを用いてセキュリティ要求を記述した手法を提案している[24)]が，要求と脅威との関連性が問題フレーム上に記述されないため，対策の十分性が問題フレームの図から直接的には把握しにくくなっている．

3.4　セキュリティに特化した手法

開発プロセス　セキュリティに特化した手法のうち，SDL，CC，SQUARE はセキュリティ開発のプロセス（手順）を規定するものである．これらの手法では要求分析の手順として，保護資産分析，脅威分析，対策抽出のステップの順に行うことが規定されている．しかし，各ステップにおける達成要求が示されるのみで，具体的な手順については規定されない．SDL や CC，SQUARE を用いる対象は，あくまでも脅威分析やセキュリティ知識を持つことが前提になっている．

脅威モデリング　脅威モデリングでは，分析者はまずソフトウェアをコンポーネントに分解し，コンポーネント間のデータフローダイアグラム（DFD）を記述する．次に DFD の中で攻撃の対象になりうる箇所を抽出し，脅威の抽出を行う．次に抽出された脅威を具体的に実現する手段に分解し（脅威ツリーの作成），実現可能性の分析と影響評価を行う．脅威モデリングは，コンポーネント間に流れるデータを保護資産と見なせば，データの入力や出力が行われる箇所において脅威を抽出することができるため，脅威抽出のための道具としては有用である．しかし，脅威モデリング手法にも下記に示すような問題点がある．

- 脅威抽出，脅威ツリーの作成にはセキュリティ知識が必要になる．
- 脅威抽出のためにはある程度ソフトウェアコンポーネントの分解が必要である．また，脅威ツリーの作成の際にも，アーキテクチャやシステム設計仕様に関する情報が前提知識として必要になる．
- DFD または脅威ツリーでは，アクタ（役割）に対応する概念がない．同じ攻撃箇所でも，攻撃者の種類（攻撃箇所，役割）によって異なる攻撃が可能になるが，その区別がされていないため，抽出漏れが発生する可能性がある．

4. 提 案 手 法

筆者らは，限られたセキュリティ知識に基づく要求の策定手法として，セキュリティ有識者と開発者の分担を明確化したアスペクト指向のセキュリティ要求策定プロセス（AOSRE）を，また AOSRE を効率的に行うための記法として，ミスユースケースに保護資産ベースの拡張を加えた AsseMis を提案する．

4.1　セキュリティ知識の差異を考慮したアスペクト指向要求策定プロセス（AOSRE）

本章では，ソフトウェア開発プロジェクトが次のような 2 つのグループで構成されることを前提とし，それぞれに担当する作業を割り当てたアスペクト指向のセキュリティ要求分析手法（AOSRE）を提案する．

(A)　セキュリティ有識者

ソフトウェア開発において想定される脅威，およびその対策について十分な知識を有する者．個々のソフトウェアのドメインに関する知識は不足していてもよい．また，ソフトウェアの開発に携わる者（(B) 開発者）でなくてもよい．

(B)　開発者

ソフトウェアを開発する担当者．開発対象のドメインに対する知識は十分であるが，セキュリティ知識は不足している可能性あり

また，ステークホルダの中で上記 (A)，(B) 以外の者（重要な決定事項に対する権限を持つ顧客，責任者など）も存在するが，合意を得る対象である以上の積極的な役割は負わないため，本提案プロセスにおいては作業は割り当てられない．上記の構成を前提におく根拠は次のとおりである．2 章で述べたように，ソフトウェア開発において，セキュアな開発に必要なセキュリティ知識を有する者の数は不足する傾向にあるため，ソフトウェア開発においては，外部から有識者を導入しなければならない場合もある．その場合，導入された有識者

は開発対象ソフトウェアのドメインの専門家ではないため，ドメイン知識は不足している可能性がある．

AOSRE はセキュリティ知識，ドメイン知識の保有者が上記 (A)，(B) のように分離していることを前提におき，セキュリティ要求分析，策定作業をセキュリティ知識，ドメイン知識に応じて作業可能なように，責任範囲の明確化を目的とする．なお，セキュリティ有識者がドメイン知識に通暁していたり，開発者がセキュリティ有識者であったりする場合は，本提案手法においては，他方に割り当てられた作業を兼任してもよいものとする．

ISO/IEC TR 15446[25] に述べられているように，CC における基本設計書である PP（Protection Profile）や ST（Security Target）を作成するためには何が脅威かを識別する必要があり，脅威の識別のためには，保護資産の特定が必要となる．また，システムの（セキュリティ以外の）機能が先に定義されていれば，機能で扱われるデータなどが保護資産になるため，保護資産の識別が容易になる．したがって，通常の機能要求とセキュリティ要求を別に定義することを前提にすると，一般的には**図 2** に示すような手順となる．筆者らはこの手順を実際のソフトウェア開発の要求分析で行い，各ステップに要求される知識の分類を行った．

(1)　機能の定義
ソフトウェアの（セキュリティ以外の）一般的な機能要求の定義を行う．対象になるドメインの知識が必要．

(2)　保護資産の抽出，評価
定義した機能から，資産（機能および機能が扱うデータ）を抽出する．抽出した資産からさらに，セキュリティ上保護する必要があるものを「保護資産」として抽出する．資産は各機能にかかわるものであるため，主としてドメイン知識を必要とする．しかし，資産のうち，それを保護資産にすべきかどうかは，資産に様々なセキュリティ的脅威を想定してみて判断するため，セキュリティ知識も必要になる．

(3)　アクセス権の定義
この作業もセキュリティ知識，ドメイン知識の双方を必要とする．

(4)　脅威の抽出
主にセキュリティ知識，特に具体的攻撃などの脅威にかかわる知識が必要．

(5)　脅威の評価
対策の必要性を判断するために，脅威の実現可能性や保護資産への影響の評価を行う．この作業は主にセキュリティ知識を必要とするが，脅威評価にステークホルダのポリシなどが基準として用いられることもあるため，場合によってはドメイン知識も必要とする．

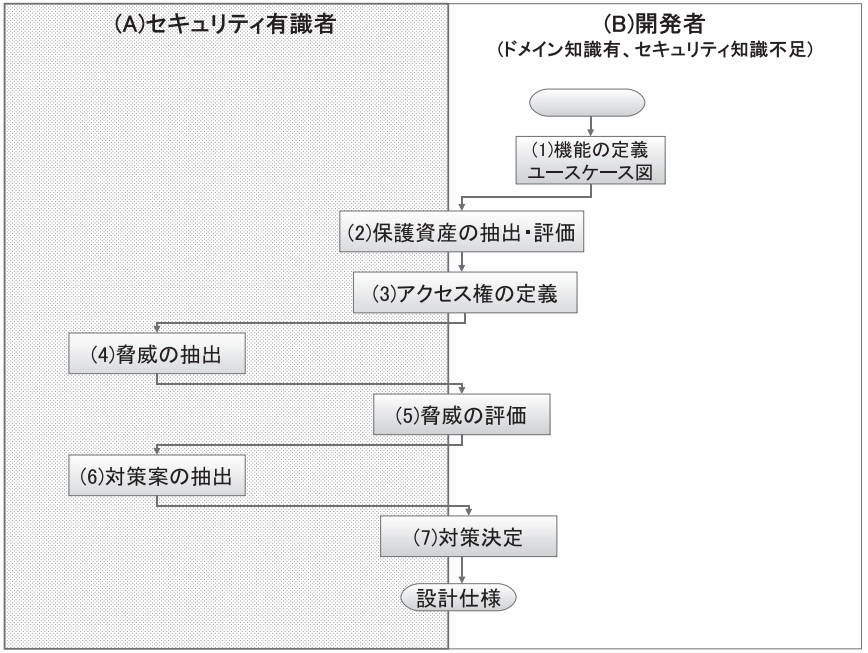

図 2　提案手法の手順
Fig. 2　The procedure of the proposed method.

(6)　対策案の抽出
抽出された脅威に対して，想定される対策案を抽出する．この作業は主にセキュリティ知識を必要とする．

(7)　対策の決定
抽出された対策案から，脅威評価に基づき実際に行う対策を決定する．(5) 同様，決定にはセキュリティ知識が必要だが，最終的な決定権にはドメイン知識も必要になる．

なお，上記の手順の最終成果であるセキュリティ要求については，以下の要素で構成されるものと定義する．

- ソフトウェアに存在する保護資産の一覧

- 各保護資産に対する脅威の一覧
- 各保護資産のそれぞれの脅威に対する対策の一覧

以上の分析結果をふまえ，AOSRE では主にセキュリティ知識を必要とするステップを (A) セキュリティ有識者の分担，ドメイン知識を必要とするステップを (B) 開発者の分担，双方の知識が必要なステップを (A)，(B) の共同作業とした．AOSRE における要求策定のための手順を図 2 に示す．図中で，左半分の (A) の領域に属するステップは (A) の分担，右半分の (B) の領域に属するステップは (B) の分担，中央に位置するステップは (A)，(B) 双方の共同作業であることを示す．

AOSRE は，セキュリティ要求策定のための手順から，セキュリティ知識を要する作業を分離することで，少数のセキュリティ有識者が，すべての要求策定に関与しなくても済むことを可能にする．しかし，AOSRE においては，脅威や対策の抽出作業自体は従来のままであるため，AOSRE のみでは脅威分析作業の軽減はできない．以下では，AOSRE の中で利用することで，セキュリティ有識者による脅威分析，対策の抽出を効率化する記法 “AsseMis” を提案する．

4.2　脅威分析，対策抽出における保護資産ベースミスユースケース（AsseMis）

前章で述べたように，脅威分析，対策の抽出を行う既存手法としてはミスユースケースと脅威モデリングがあり，どちらの手法にもそれぞれ長所，短所がある．ミスユースケースは結果の視認性に優れるが，分析に必要な資産やアーキテクチャ情報が欠けている．一方脅威モデリングには資産やアーキテクチャの概念があり分析は比較的容易であるものの，対策も含めた結果の確認の視認性ではミスユースケースに劣る，そこで，提案手法では双方の手法の利点を活用するため，ミスユースケースを独自に拡張し，保護資産と簡易的なアーキテクチャの概念を導入した「保護資産ベースミスユースケース手法（Asset based Misuse case approach（AsseMis））」を新規に提案する．AsseMis は上記の他にも脅威，対策抽出を容易にするため，従来のユースケース図やミスユースケース図に対しいくつかの記法を新規に追加する拡張を行っている．

4.2.1　ね　ら　い

- 保護資産情報に基づく保護資産ベースの分析による，脅威抽出の効率化
  文献 25) が述べているように，脅威の識別にはまず，保護資産を識別する必要がある．しかし現状のユースケース図やミスユースケース図には保護資産を表現する要素がない．保護資産を識別しそれを表現することにより，保護資産ベースの脅威分析を可能にしたい．

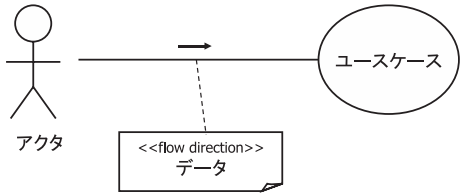

図 3　データフローを表記する拡張モデル
Fig. 3　The model extension for describing data flow.

- 要求分析段階レベルの粗いアーキテクチャに基づく脅威識別，評価の容易化
  従来のユースケース図（ミスユースケース図）は機能を要求レベルで表現するものであるため，もともとアーキテクチャの概念は表現されてない．しかし，そのままでは抽出した脅威の可能性の評価（リスク評価）はまったくできなくなってしまう．そのため，要求分析段階でも明確になっている粗いレベルのアーキテクチャ情報を利用し，それを表現することによって脅威評価の助けとしたい．
- セキュリティ有識者自身による，脅威に対する対策抽出の十分性の確認の容易化
  従来のミスユースケース図では，脅威に対策を接続させることにより，脅威に対する対策の有無を確認することで漏れがないか確認できるようになっている．しかし，同一の脅威でも，発生する箇所により対策が異なる場合がある．このような場合，1 個のミスユースケース（脅威）に複数の対策ユースケースが接続することになる．したがって，脅威に対する対策が 1 つ存在することがミスユースケース図の接続によって関連付けられたからといって，それで対策が十分であるとは限らない．このように，従来のミスユースケースの記法では脅威，対策抽出の十分性の確認は容易ではないので，これを容易にしたい．

4.2.2　拡 張 内 容

(1)　ユースケース図への保護資産表記の追加
ユースケース図上で保護資産ベースの脅威分析を可能にするため，AsseMis では，従来のユースケースにはない保護資産の表記を新規に追加する．具体的には，次の 2 点の追加を行う．

- データフロー表記の追加（拡張 (I)）
  各ユースケースで扱うデータを，**図 3** に示すようにユースケース図に新規に追加する．データは UML のノート形式でユースケースとアクタを関連づける線

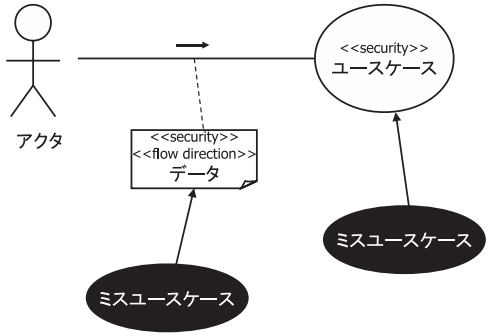

図 4　保護資産と関連づけるミスユースケースの拡張モデル
Fig. 4　The misuse case extension for describing assets.

上に接続するものとする．データをユースケースではなく，線に接続させる理由は，データをユースケース，アクタ双方に関連づけることによって，データへのアクセス権を明確にさせるためである．また，データの流れの方向を ⟨⟨flow direction⟩⟩ ステレオタイプで示す．アクタからユースケースに入力する場合は ⟨⟨send⟩⟩，ユースケースからアクタに出力する場合は ⟨⟨receive⟩⟩ ステレオタイプで表記する[⋆1]．データだけでなく，データの流れの情報を記述させる目的は，セキュリティにおいてはソフトウェアに対する入力データや出力データに対して行う攻撃がよく知られており，入力の場合，出力の場合を区別しておけば，想定される脅威や対策の抽出が容易になるためである．たとえば，SQL インジェクションやバッファオーバフロー，クロスサイトリクエストフォージェリ（CSRF）攻撃などは入力データを利用する攻撃であるため，データの入力がある場合にこれらの攻撃が脅威候補となりうる．

- セキュリティプロパティの追加（拡張 (II)）
  保護すべき保護資産を識別するためには，それがセキュリティ上どのように重要かを示す必要がある．そこで AsseMis では，ユースケース図上のデータおよびユースケースに，セキュリティプロパティを表すステレオタイプを用いてこの情報を追加できるようにする．ステレオタイプは，次に示す 4 種類とし，**図 4** の ⟨⟨security⟩⟩ ステレオタイプの位置に記述する．セキュリティプロパティは複数記述してもよい．
  - ⟨⟨C⟩⟩ 要機密性
    対象に対して，許容されたアクタ以外のアクセスを禁止する必要性がある場合に付与
  - ⟨⟨I⟩⟩ 要完全性
    プロパティを付与した対象の意図しない改竄を禁止する必要性がある場合に付与
  - ⟨⟨A⟩⟩ 要可用性
    プロパティを付与した対象の機能妨害や悪用を禁止する必要性がある場合に付与
  - ⟨⟨P⟩⟩ 要プライバシ性
    要機密性に加え，許容されたアクタでも，その対象のアクセス権を保有する者以外へのアクセスを禁止する必要性がある（利用者自身の個人情報など）場合に付与

  上記プロパティを有するデータ，およびユースケースを保護資産とする．

（2）　ミスユースケースと保護資産の対応付け（ミスユースケースの接続先の変更）（拡張 (III)）
Sindre，Opdahl のミスユースケース[3]（図 1 参照）では，ミスユースケースは矢印によって通常のユースケースに接続されている．これは，そのユースケースが表す機能において，ミスユースケースが表す脅威が存在することを表している．しかし，保護資産ベースの分析という観点では，実際に脅威に晒されるのは保護資産そのものであり，脅威を表すミスユースケースは保護資産そのものに結びつける方が自然である．AsseMis では，データおよびユースケースが保護資産になりうる．したがって，AsseMis では従来のミスユースケースの記法を変更し，図 4 に示すようにユースケース以外にデータにも接続できるようにする．

（3）　アーキテクチャ情報（拡張 (IV)）およびミスユースケースの依存関係（拡張 (V)）の追加
従来のユースケース，および Sindre，Opdahl のミスユースケース[3]（図 1 参照）にはアーキテクチャを表す概念がない．そこで AsseMis では，要求分析段階において，アーキテクチャ情報がある程度提供される場合，脅威を表すミスユースケースにそ

⋆1 図 3 では視認性を高めるため矢印を補足しているが，矢印表記はなくてもよい．

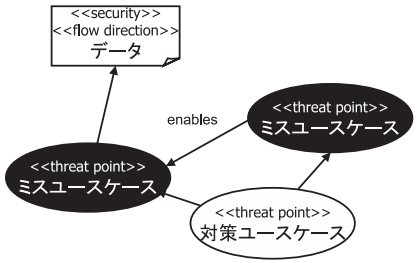

図 5　脅威の位置と依存関係を記述する拡張モデル
Fig. 5　The model extension for describing threat points and dependencies.

の脅威が発生する可能性のあるアーキテクチャ上の箇所を**図 5** におけるミスユースケースの ⟨⟨threat point⟩⟩ の位置にステレオタイプで追加できるようにする．たとえば，対象とするシステムがクライアント–サーバ構成の場合，クライアントが発生箇所になる脅威には ⟨⟨client⟩⟩，通信路が発生箇所になる脅威には ⟨⟨channel⟩⟩，サーバが発生箇所になる脅威には ⟨⟨server⟩⟩ と記述するようにする．1 つの脅威に対して複数の発生箇所を記述してもよい．

対策の脅威との対応付けを明確にし，対策漏れを発見しやすくするために，対策を表すユースケースにも，ミスユースケースと同様に脅威が発生する可能性のあるアーキテクチャ上の箇所を，図 5 の対策ユースケースの ⟨⟨threat point⟩⟩ の位置にステレオタイプ形式で記述する．

スタンドアローンのシステムのように発生箇所を区別する必要のない場合や，不明の場合は，この脅威の発生箇所の記述はなくてもよい．

また，脅威の発生する可能性を評価するために，脅威どうしの依存関係の表記を新規に追加する．ある脅威を実行可能にする手段として，別の脅威（または攻撃）が想定される場合，図 5 に示すように，後者のミスユースケースから前者のミスユースケースに矢印を引き，“enables” と表記する．

(4)　ミスアクタの拡張（拡張 (VI)）Sindre，Opdahl のミスユースケース[3]（図 1 参照）では，ミスアクタは 1 種類しか定義されていない．

しかし，セキュリティの見地からは，アクタをさらに区別した方が望ましい場合がある．たとえば，個人のプライバシ情報を扱う場合，それが「利用者」という同じアクタどうしでも，プライバシ情報の保有者以外が参照すれば脅威（情報漏洩）になりうる．また，アクタ自身による攻撃や，アクタが意図せずに，または騙されて攻撃を実行してしまう場合（Web におけるクロスサイトスクリプティング（XSS）やクロスサイトリクエストフォージェリ（CSRF）など）を区別して表記できる方が望ましい．そこで，AsseMis では**図 6** に示すように，ミスアクタをさらに (1) 本人（対象としているアクタのインスタンス自身）と，(2) 本人以外（対象としているアクタと同じアクタだが，異なるインスタンス）の 2 種類に区別した表記を追加する．

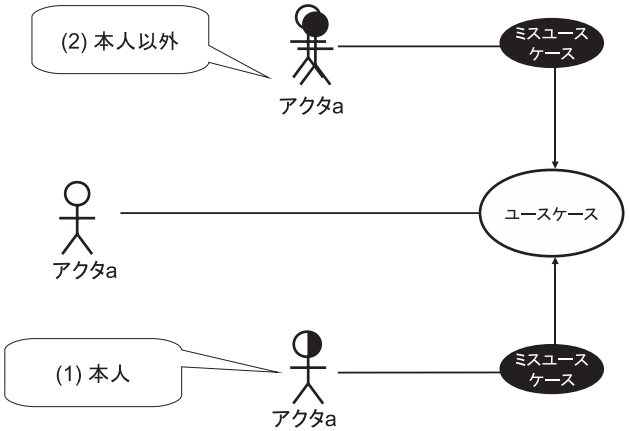

図 6　ミスアクタの分類追加
Fig. 6　The model extension for describing multiple types of mis-actor.

4.3　AsseMis を用いた AOSRE

AOSRE の各ステップにおいて，AsseMis を利用することで，AOSRE による脅威，対策抽出をより効率的に行うことが可能となる．その利用手順を次に示す．

(1)　機能の定義（(B) 開発者による作業）
UML のユースケース図を用いて，アクタ，ユースケースを記述する．

(2)　保護資産の抽出，評価（(A) セキュリティ有識者，(B) 開発者双方による作業）
ユースケースで用いるデータを抽出し，そのフローの情報とともに拡張 (I) の形式を用いてユースケース図に追加する．その後，各データおよびユースケースに対しセキュリティ的な重要性を検討し，セキュリティプロパティを拡張 (II) の形式によりステレオタイプとして付与する．

(3)　アクセス権の定義（(A) セキュリティ有識者，(B) 開発者双方による作業）

アクタとユースケース間，およびアクタとデータ間のアクセス権を整理する．(2) のユースケース図を記述した時点で，アクセス権の定義はある程度できている．たとえば，あるアクタにユースケースが接続されていれば，アクタはそのユースケースに対して実行権があると解釈できる．また，アクタとユースケース間にデータが接続されている場合，そのアクタはデータへのアクセス権があることを示す．また，参照（read）権か更新（write）権かはフローの方向によって判断することができる．フローの方向が ⟨⟨send⟩⟩ すなわちアクタからユースケースに流れるのであれば更新権，⟨⟨receive⟩⟩ すなわちユースケースからアクタに流れるのであれば参照権である．
上記の手順でアクセス権を整理し，最後にアクセス権をすべてチェックし，アクセス制御は適切か，必要な制御に抜けがないかどうかを確認する．その際，図だけでは表記できない細粒度のアクセス権が要求されている場合は，補足として文章などで記述しておく．

(4)　脅威の抽出（(A) セキュリティ有識者による作業）
(2) で記述したユースケース図の各保護資産（セキュリティプロパティが付与されたデータまたはユースケース）に，拡張 (III) の形式によりミスユースケースを追加する．追加にあたっては，保護資産に付与されたセキュリティプロパティの種類，データフローの方向，(3) のアクセス権定義を利用し，保護資産に被害を及ぼす可能性のある脅威を列挙していく．脅威の抽出には，たとえば文献 13) の脅威分類 STRIDE（S（なりすまし），T（改竄），R（否認），I（情報漏洩），D（DoS），E（権限昇格））を用いるのも 1 つの方法である[⋆1]．STRIDE を用いる場合では，機密性に対しては情報漏洩，完全性に対しては改竄，可用性に対しては DoS，否認のミスユースケースをそれぞれ追加する．

(5)　脅威の評価（(A) セキュリティ有識者，(B) 開発者双方による作業）
抽出された各ミスユースケースに対して，ミスアクタの種類，前提となるアーキテクチャ情報に基づき，実際にその脅威が発生しないかどうかを検討する．手順としては，まず各ユースケースに対して，すべて種類のミスアクタ，およびすべての脅威の発生箇所となるポイントを列挙し，ミスユースケース上に拡張 (IV)，拡張 (V) の形式を用いて列挙する．次に，ある脅威を可能にする別の脅威が考えられる場合は，拡張 (V) の形式により，依存関係付きでミスユースケースを追加する．この 2 次的な脅威抽出においても STRIDE 分類を用いることができる．ここで 2 次的脅威の候補になりうるのは，S（なりすまし）や E（権限昇格）など，保護資産に直接害を及ぼすのではないが，他の脅威を可能にする手段としては用いられるものである．2 次的な脅威の抽出を行ったうえで，それぞれのミスアクタや脅威の発生箇所をもとに可能性を検討する．可能性のある攻撃が存在しないもの，アクタの性質やアーキテクチャ上除外していいもの，物理セキュリティにより守られている，もしくは運用により守られているミスユースケースは図から削除していく．このようにして，残ったものが「対策が必要」と評価された脅威となる．

(6)　対策案の抽出（(A) セキュリティ有識者による作業）
脅威（ミスユースケース）の抽出，評価を行ったミスユースケース図中の各ミスユースケースに対して，対策になりうるユースケースを追加する．対策案の抽出は主にセキュリティ知識と脅威の発生箇所，手法，ミスアクタの種類に基づき行うが，基本的に 1 つの脅威発生箇所に対して，最低 1 つの対策が関連づけられるようにする．

(7)　対策の決定（(A) セキュリティ有識者，(B) 開発者双方による作業）
抽出された対策案をもとに，対策の実現可能性，セキュリティポリシ，運用などを検討し，最終的に行う対策を決定する．

5. 評　　価

本章では，提案したミスユースケースの拡張（AsseMis），および AsseMis を利用したプロセス（AOSRE）の有効性について，具体的な攻撃事例，実際のシステムへの適用事例をもとに評価する．

5.1　AsseMis による拡張の効果に対する評価

プライバシ情報に対する脅威，対策の表現を具体例として評価する．あるプライバシ情報（他の利用者は参照できない情報）を含む情報を Web 経由でブラウザで参照させる機能について，セキュリティ要求を Sindre，Opdahl のミスユースケース[3] 形式で表記した図が 3 章の図 1 である．これに対し，同じ対象を AsseMis で表記したものを**図 7** に示す．なお，図 7 では，対策ユースケースと関連する矢印を青色で表記しているが，説明上対策ユースケースと他のユースケースの区別を明確にするために色分けしており，AsseMis 記法で色を区別することは必須ではない．

脅威，対策の抽出，評価の点で，Sindre，Opdahl のミスユースケース[3] に比べ，AsseMis

⋆1 筆者らは，これまでに行った Web アプリケーションの脅威分析においてもすでに CIA のセキュリティ分類と STRIDE を用いており，それらによって十分に脅威，対策が抽出できることは確認済みである．

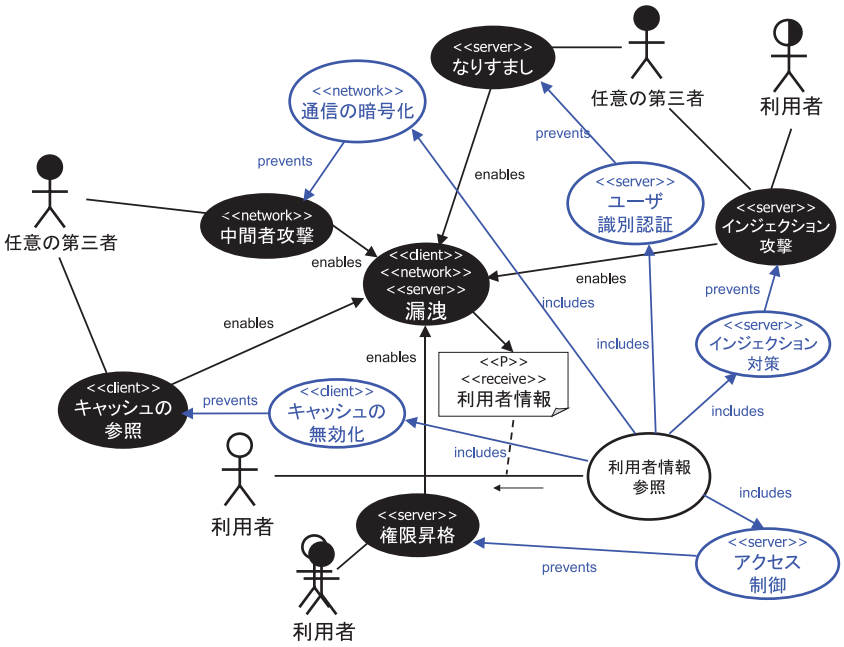

図 7　プライバシ情報の参照機能：AsseMis による表記
Fig. 7　An example of AsseMis: the function of viewing private information.

は次の点で有利であり，4.2.1 項で述べたねらいを満足する効果が得られることが確認できる．

- 保護資産情報に基づく保護資産ベースの分析による，脅威抽出の効率化
  - 保護資産の表現
    Sindre，Opdahl のミスユースケース[3]（図 1 参照）には保護資産の表記がないため，図から直接保護資産を読みとれない．一方，AsseMis（図 7）では，セキュリティプロパティ（⟨⟨P⟩⟩）の付与されたデータ（「利用者情報」）が保護資産であることが図から明確に分かる（4.2 節の拡張 (I)，(II) による効果）．
  - 脅威の表現
    Sindre，Opdahl のミスユースケース[3]（図 1 参照）では，前述のように保護資産の表現がないため，ミスユースとして表記されている脅威がどの保護資産に対応するのかは図からは直接読みとれない．一方，AsseMis（図 7）におけるミスユースケース「漏洩」は，データ「利用者情報」に接続しているため，この機能における脅威が「利用者情報という保護資産に対する漏洩」であることが図から読みとることができる（4.2 節の拡張 (III) による効果）．
- 要求分析段階レベルの粗いアーキテクチャに基づく脅威識別，評価の容易化
  - アーキテクチャ情報
    Sindre，Opdahl のミスユースケース[3]（図 1 参照）はユースケースをもとにしているため，アーキテクチャについての情報は表れていない．しかし，このシステムは Web アプリケーションであることが事前に分かっているため，サーバ，クライアント，およびネットワークの 3 つが脅威の発生箇所候補になることが分かる．そこで，AsseMis（図 7）では，それぞれの脅威発生箇所に対応する ⟨⟨client⟩⟩，⟨⟨server⟩⟩，⟨⟨network⟩⟩ ステレオタイプをミスユースケースに付与することにより，各脅威の現実的な可能性についてアーキテクチャ情報をもとに評価できるようになる（4.2 節の拡張 (IV) による効果）．
  - 具体的手段の識別による評価
    Sindre，Opdahl のミスユースケース[3]（図 1 参照）では，1 つの「漏洩」というミスユースケースがあるのみである．しかし，「漏洩」には，「なりすまし」「中間者攻撃による通信路盗聴」など具体的に実現する攻撃手段があるはずで，それらの詳細化を行わないと，「漏洩」の脅威評価を行うことも，「ユーザ識別，認証」「暗号化」などの具体的対策を識別することも困難である．一方，AsseMis（図 7）では，依存関係の追跡により「漏洩」を実現する手段として，具体的な攻撃「中間者攻撃」「キャッシュ漏洩」「インジェクション攻撃」「なりすまし」「権限昇格」があることが分かり，それぞれに対策が必要であることも分かる（4.2 節の拡張 (V) による効果）．
- セキュリティ有識者自身による，脅威に対する対策抽出の十分性の確認の容易化
  Sindre，Opdahl のミスユースケース[3]（図 1 参照）には，「漏洩」という 1 つのミスユースケースに，複数の対策が接続されている．これらの対策は前項で示したようにそれぞれ別の具体的脅威に対応するものなので，1 つの対策が別の対策の代用になるものではなく，それぞれがすべて必要なものである．しかし，ミスユースケースでは，それが表現されていないため，この中の 1 つが欠けていても，図によるチェックでは判明しない可能性がある．たとえば，図 1 中の対策の 1 つ「通信の暗号化」がセキュリティ

対策から抜けていたとする．この抜けを図でチェックしようとしても，複数の対策が脅威に連結しているので，図からすぐには何が欠けているかは判明しない．

一方，AsseMis では，次のような手順でチェックを行うことで，機械的に対策漏れをチェックできる．

(1)　対象の脅威ミスユースケースを可能にする（enabled 矢印で指示されている）ミスユースケースに対する対策ユースケースがすべて存在するか．

(2)　対象の脅威ミスユースケースに付与されているすべての発生箇所ステレオタイプについて，その発生箇所ステレオタイプが付与された対策ユースケースが存在するか．

Sindre，Opdahl のミスユースケース[3] の例と同じように，図 7 において，「通信の暗号化」が抜けていた場合，上記の手順に従って図をチェックすれば，下記のユースケースが足りないことが分かる．

(1)　図 7 の「漏洩」を可能にする「中間者攻撃」ミスユースケースに対応する対策ユースケースが存在しない．

(2)　図 7 の「漏洩」に付与された ⟨⟨network⟩⟩，すなわちネットワークにおける漏洩の脅威に対応する対策ユースケースが存在しない．

このようにして，脅威に対する対策漏れを容易に検証できる（4.2 節の拡張 (V) による効果）．

また，このシステムでは利用者情報はプライバシ情報であるので，本人以外にアクセスさせないためのアクセス対策が必要である．しかし，Sindre，Opdahl のミスユースケース図[3]（図 1）では，ミスアクタは同じ「利用者」としか表現されていないので，「本人以外への漏洩」という脅威は表現できない．その場合，対策でそれが漏れてしまう可能性もある．一方，AsseMis では，本人以外のアクタに対応するミスアクタ表現があるため，それを用いて図 7 のように権限昇格のプレーヤとして「本人以外のミスアクタ」を表現することができる．したがって，対策を考慮する際に，「本人以外の利用者への漏洩を防ぐためのアクセス制御対策」が必要であることが図から読みとれるようになる．このようにして，漏れの可能性を減じることができるようになる（4.2 節の拡張 (VI) による効果）．

5.2　実システム開発への適用による AOSRE の評価

筆者らは，AOSRE と AsseMis を組み合わせた手法を，実際の Web アプリケーション開発のセキュリティ要求策定に適用した．対象のアプリケーションは 6 つで，いずれもインターネットを経由して利用する形式の Web アプリケーションである．各アプリケーションの概要を以下に示す．

(a)　プライバシ情報を含む参照機能を持つアプリケーション（ユースケース数：1）．

(b)　特定の利用者を対象とした申請機能を持つアプリケーション（ユースケース数：2）．

(c)　公開情報を提供するアプリケーション．ただしインターネットを経由したメンテナンス機能を含む（ユースケース数：4）．

(d)　組織内で共有する情報を参照および登録，更新する機能を持つアプリケーション（ユースケース数：5）．

(e)　有償の情報を購入操作を行うことにより参照可能にする機能を持つアプリケーション（ユースケース数：6）．

(f)　会議への参加者，関係者の登録，参照機能を持つアプリケーション（ユースケース数：9）．

5.2.1　効率化に関する評価

AOSRE と AsseMis を組み合わせた手法を Web アプリケーション開発のセキュリティ要求策定に適用した結果について，従来作業とのセキュリティ作業量（セキュリティ有識者が行った作業の作業量）の比較を行った．比較対象として，セキュリティ要求分析についての過去の計測データはなかったため，既存アプリケーションのセキュリティ診断で用いられていた脅威分析の作業量のデータを用いた．従来のセキュリティ診断で用いられていた脅威分析の手順を次に示す．

(1)　診断者（セキュリティ有識者に相当）は通常対象のアプリケーションについての知識はないため，最初に対象アプリケーションの機能の概要について，開発者から聴取することで理解する（セキュリティ有識者，開発者の対面による作業）．また，この作業を通じて何がアプリケーションにとっての保護資産なのかを診断者が理解する．

(2)　診断者は，引き続き開発者から聴取しながら，対象アプリケーションの詳細なアーキテクチャ情報を理解し，脅威モデリングの DFD 図などを用いて記述する（セキュリティ有識者，開発者の対面による作業）．

(3)　(1)，(2) で得られた情報をもとに，脅威を抽出する（セキュリティ有識者，開発者の対面による作業）．

(4)　抽出した脅威の発生可能性について，(2) のアーキテクチャ情報に基づき評価する（セキュリティ有識者，開発者の対面による作業）．

(5)　抽出された脅威情報をもとに，診断者は設計書，ソースコード分析，脆弱性監査など

表 1　AOSRE のセキュリティ作業量
Table 1　Workload with AOSRE.

アプリケーション	ユースケース数（個）	保護資産数（個）	セキュリティ作業量（人時間）	(B) 開発者との対面での共同作業量（人時間）	共同作業回数（回）
(a)	1	1	2.5	1	1
(b)	2	1	10	4	2
(c)	4	7	14	6	2
(d)	5	8	14	6	2
(e)	6	11	18.5	4.5	3
(f)	9	16	14	4	2

を用いて脆弱性の検査を行う（セキュリティ有識者単独の作業）．

(6)　抽出された脆弱性情報をもとに，必要な対策を検討し決定する（セキュリティ有識者単独の作業）．

従来の脅威分析作業と AOSRE との相違点は，(2) のアーキテクチャ記述，(3) の脅威抽出において，セキュリティ有識者と開発者の対面作業を必要とする点，および，セキュリティ有識者が単独で行う (5) の脆弱性検査が増えている点である．従来の脅威分析の対象になったアプリケーションは，今回 AOSRE の計測に用いた前述の 6 つとは (e) を除き異なる，ユースケース数 1〜12 の 6 つのアプリケーションである．(e) に関しては，同一のアプリケーションに対して，別のチームがそれぞれ診断と要求分析を並行して行った．(e) はすでに完成したアプリケーションだが，セキュリティ要求が明確でなかったため，AOSRE による要求分析を通して要求の明確化を試みている．

AOSRE のセキュリティ作業量を**表 1** に，従来の作業のセキュリティ作業量を**表 2** に示す[⋆1]．また，両者のセキュリティ作業量についてユースケース数を横軸に，作業量を縦軸にして比較したグラフを**図 8** に示す．なお，上記で述べたように，ユースケース数はほぼ同じだが一部を除き異なるアプリケーションについて比較を行っている．

従来の診断では，セキュリティ有識者の作業量がユースケース数に比例して大きく増加しているのに対し，AOSRE の作業量は相対的に小さく，ユースケース数が増加しても 10〜20 人時間の範囲にとどまっており，大きな変化がないことが分かる．従来の診断作業による作業量が AOSRE に対して大きい要因として，従来の診断作業は既存アプリケーションの分析作業であるため，セキュリティ作業量の中には完成したアプリケーションの脆弱性監

⋆1 従来の診断では，保護資産の数は明示的に抽出されないため，表 2 では保護資産数のデータはない．

表 2　従来診断のセキュリティ作業量
Table 2　Workload with conventional security assessment.

アプリケーション	ユースケース数（個）	セキュリティ作業量（人時間）	(B) 開発者との対面での共同作業量（人時間）	共同作業回数（回）
(g)	1	13.5	7.5	1
(e)	6	43	25	5
(h)	9	52	12	1
(i)	9	44.5	4.5	1
(j)	9	11.5	15	5
(k)	12	129	21	7

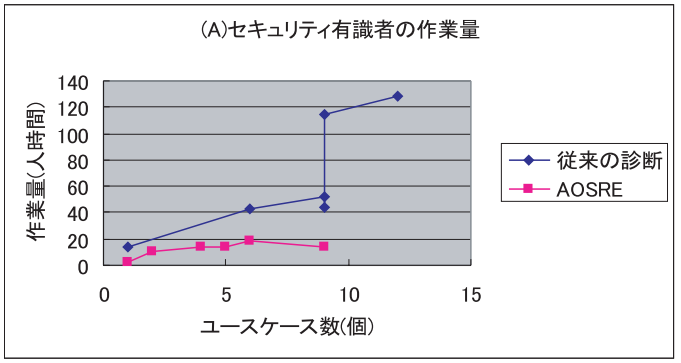

図 8　セキュリティ作業量比較
Fig. 8　Comparison of security workload.

査の作業量が含まれており，AOSRE には含まれていないことが考えられる．そこで，より比較を正確に行うため，セキュリティ作業のうち (B) 開発者と対面で行う共同作業の作業量に限定して両者を比較した結果を**図 9** に示す．

従来の診断における脅威分析作業では，対面による作業は手順の (1)〜(4) にわたるため，多くの時間，回数が割かれていた（DFD 記述や脅威抽出，評価の作業は特に長時間に及ぶため，回数を分割して行うことが多い）．このことにより，セキュリティ有識者は拘束される時間が多くなり，他の作業を並行して行うことが困難になっていた．一方，AOSRE では，対面による作業，回数ともに減少していることが分かる．これにより，拘束時間も短くなり，セキュリティ有識者を他のアプリケーションの作業に割り当てることがより容易になったと考えられる．なお，AOSRE では対面による作業を保護資産の抽出，脅威の評価，

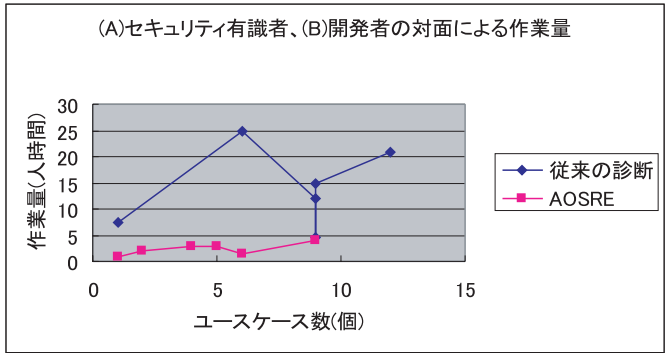

図 9　(B) 開発者との対面での共同作業量比較
Fig. 9　Comparison of security workload (face-to-face).

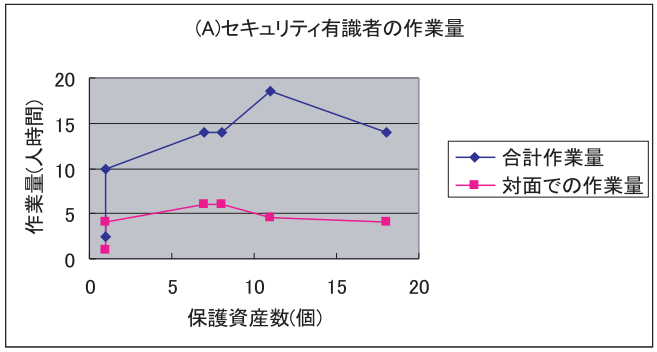

図 10　セキュリティ作業量（対保護資産数）
Fig. 10　Comparison of security workload by the amount of assets.

対策の決定の 3 回としているが，実際の適用では脅威の評価や対策の決定ではメールによるやりとりのみで，対面作業は省略され，2 回または 1 回の場合もあった．

今回適用したアプリケーションはすべてユースケース数が 10 以下のものだった．アプリケーション規模がより大きくなった場合の AOSRE の作業量への影響を調査するために，ユースケース数のほかに保護資産数を横軸にしたグラフを**図 10** に，抽出された脅威数，対策数を横軸にしたグラフを**図 11** に示す（抽出された脅威数，対策数のデータについては，後述の**表 3** を参照）．

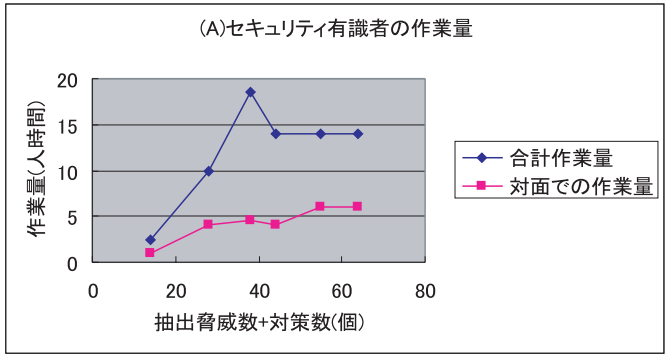

図 11　セキュリティ作業量（対脅威・対策抽出数）
Fig. 11　Comparison of security workload by the amount of identified threats and countermeasures.

表 3　AOSRE による抽出結果と後工程での診断結果
Table 3　The result of threat identification with the result of security assessment later.

アプリケーション	AOSRE で抽出した脅威数（個）	AOSRE で抽出した対策数（個）	1 回目（設計時）診断で発見されたリスク数（個）	2 回目（監査）診断で発見された脆弱性数（個）
(a)	6	8	（未完）	（未完）
(b)	14	14	1	1
(c)	36	19	2	0
(d)	40	24	（未完）	（未完）
(e)	24	14	0	0
(f)	28	16	（未完）	（未完）

アプリケーションの規模の尺度は，ユースケース数や保護資産数で表現することができるが，AOSRE では，ユースケース数や保護資産数により作業量の大きな変動はみられない．一方，脅威，対策の抽出数が少ない場合はこれらの増加による作業量の増加傾向が見られる．作業量がユースケース数や保護資産数にあまり影響されない理由は，作業の途中で，複数のユースケースの中に同様のパターンが頻出したため，1 回目に抽出した脅威，対策の AsseMis 図を再利用していることが増えているためと考えられる．実際に Web アプリケーションにおいてはアーキテクチャの形態が限定されるため，抽出される脅威，対策も典型的なものが多い．これらは保護資産やデータの流れの形態によって，パターン化による再利用ができる可能性がある．以上のことから，AOSRE の作業量は，ユースケース数や保護資産数で表現されるアプリケーションの規模そのものよりも，抽出される脅威や対策の数により

影響を受けることが推測される．対象が Web アプリケーションなどで，想定される脅威や対策には既知のものが多い場合は，規模が増大しても作業量は大幅には増大しないのではないかと考えられる．

5.2.2　脅威，対策抽出の十分性に対する評価

AOSRE で脅威，対策抽出を行った 6 件のアプリケーション開発のうち，3 件について開発の後工程において 2 度のセキュリティ診断を行った．1 回目は設計段階におけるセキュリティ有識者による設計仕様のセキュリティ診断，2 回目は開発後に脆弱性診断ツールなどを用いて行われるアプリケーション監査である．AOSRE で抽出した脅威，対策の数，1 回目，2 回目の診断で発見されたリスク，脆弱性の数を表 3 に示す．なお，表中のアプリケーション (a)，(d)，(f) は設計段階に進んでいないため，後工程での診断は行っていない．

1 回目のアプリケーション (b) では設計で 1 件，実装で各 1 件を除き，リスク，脆弱性は発見されなかった．設計時に指摘された 1 件は，署名付き Java Applet を利用する設計仕様を採用した際に，新たにリスクが発生したものである．この件については，要求分析段階では署名付き Java Applet のアーキテクチャを採用することが決定していなかったため，AOSRE では抽出されなかった．しかし，要求分析段階では実装に用いるアーキテクチャのすべてが明らかにすることは困難であり，本提案手法においてもアーキテクチャに依存するすべての脅威の抽出は対象としていない．また，署名付き Java Applet を採用した場合でも，安全に実装すれば脆弱性を排除することは可能なので，このリスクは要求段階で抽出すべきリスクからは除外してよいと考える．また，アプリケーション (b) において実装後監査で発見された 1 件は，利用者に指示する暗号化，および署名に用いる証明書のインストール手順の，アプリケーション画面上の説明に誤りがあるというものだった．これも，説明文を修正すればよいので，AOSRE が定めたセキュリティ要求に欠落があるという問題ではないといえる．また，アプリケーション (c) でも設計時にリスクが 2 件見つかっているが，これらは，アプリケーションが行っている ID/パスワード認証のパスワードの仕様では，強度が十分でなく，総当たり攻撃に対して弱いという点に関するものである．この問題に関しては，AOSRE ではセキュリティ要求として，次の脅威と対策案が抽出され，ステークホルダ間で合意されている．

脅威：総当たり攻撃によるなりすまし，およびそれによる機密データの漏洩および改竄

対策案：総当たり攻撃に耐えうるパスワード強度の確保

したがって，設計段階で発見された問題は，セキュリティ要求に反するものであると認識され，設計仕様は修正された．その結果，実装後の監査では問題は発見されなかった．

以上のように，AOSRE の実際のアプリケーション開発への適用では，次の 2 点が効果として確認された．

- 分析段階で AOSRE を行った場合，後工程では要求段階に遡って要求を修正する必要のあるような問題は発見されなかった．すべて，発見された工程以降で対応可能なものであった．
- 分析段階で AOSRE によって定義されたセキュリティ要求についてステークホルダ間で合意しておくことで，後の工程で発見された，セキュリティ要求に違反する仕様については，要求に反するという理由で修正させることが可能になった．

また筆者らは，開発前への適用のほかに，過去にセキュリティ事故（インシデント）が発生してしまったアプリケーションについて，仮に要求分析段階で AOSRE を用いていたら，原因となる脅威を抽出し，防止する要求を定義できていたかを検証した．

- 強制ブラウジング攻撃を受けた Web アプリケーションの場合
  このアプリケーションは，利用者個人向けに，利用者自身の情報を Web で提供する機能を有しており，その中にはプライバシ情報も含まれていた．しかし，アプリケーションで特定の操作を行うことで，他人の情報の参照が可能になってしまう脆弱性があり，その脆弱性をついた強制ブラウジング攻撃を受けてしまった．この脆弱性は，利用者に対して他人の（本人以外の）プライバシ情報を参照できないようにするアクセス制御処理が必要であり，その必要性が要求定義時に認識されていることが必要だった．このアプリケーションに対し，要求分析段階で本提案手法（AOSRE）を適用した場合，まず AOSRE の「(2) 保護資産抽出」ステップで，プライバシプロパティを持つデータの参照が抽出され，⟨⟨P⟩⟩，⟨⟨receive⟩⟩ ステレオタイプを付与された保護資産として AsseMis 形式で記述される．次に，AOSRE の「(4) 脅威の抽出」ステップにおいて，プライバシデータに対応する脅威として「漏洩」が識別され，ミスユースケースとして追加される．「(5) 脅威の評価」ステップにおいては，「漏洩」に対し，AsseMis が提供するミスアクタを網羅的に適用した結果，その 1 つである図 6 の「(2) 本人以外のアクタ」による漏洩の認識が可能となり，次の「(6) 対策案の抽出」ステップで必要な対策（アクセス制御）を抽出できる．AOSRE を用い，この機能に対し要求定義した結果は先に示した図 7 のとおりになる．
- SQL インジェクション攻撃を受けたアプリケーションの場合
  対象のアプリケーションは，やはり利用者情報を Web で提供する機能を有していたが，運用中に SQL インジェクション攻撃を検出した．幸い，有効な攻撃コマンドではなかっ

たため，被害には至らなかったが，攻撃後にプログラムを検査した結果，SQL インジェクション脆弱性があることが発覚した．SQL インジェクション脆弱性は実装におけるセキュリティホールであるが，それを防ぐには開発の上流段階でインジェクションがシステムのデータ保護資産の漏洩，破壊につながる脅威であることを認識し，要求，仕様として定義することが必要になる．この脅威も AOSRE を適用することで，要求定義時に識別，対策の抽出を行うことが可能になる．まず，機能は前項のアプリケーションとほぼ同様なので，「(4) 脅威の抽出」における「漏洩」ミスユースケースまでの過程は前項と同じである．また，対象アプリケーションは Web アプリケーションなので，脅威の発生箇所としてクライアント，ネットワーク，サーバを示すステレオタイプが「漏洩」ミスユースケースに追加される．次に，「(5) 脅威の評価」において，想定する脅威発生箇所ごとに脅威を実現する具体的手段の識別，評価が行われる．Web アプリケーションにおいては，サーバ側におけるデータ漏洩や改竄の手段としては，インジェクション攻撃がよく知られている．このため，「漏洩」の「サーバ」発生箇所の評価において「インジェクション攻撃」を抽出することができる（図 7 参照）．このように，要求分析段階においてもインジェクション攻撃を「保護資産に対する脅威」としてとらえ，対策案を定義することが本提案の AOSRE では可能になる．

6. ま と め

本論文では，セキュリティ要求を分析段階で効率的に定義可能にする要求分析手法を新規に提案した．提案手法は，セキュリティ有識者と一般の開発者それぞれが担当する作業，共同で行う作業を明確にしたアスペクト指向定義手順（AOSRE）および，脅威抽出・評価，対策抽出において従来のミスユースケース記法に新規に保護資産，アーキテクチャの概念を加えた記法（AsseMis）で構成される．

AOSRE は，ドメイン有識者，セキュリティ有識者が必要に応じて個別に作業できるようにするため，要求定義作業をセキュリティ，ドメイン各知識の必要性に基づき分解したが，これらは既存の他の手法にはないものである．また，AsseMis は，脅威識別に必要な詳細情報（アーキテクチャ，具体的攻撃手法，ミスアクタのバリエーション）を提供することで，脅威識別，評価の効率性，網羅性を向上させることが可能になると考えられる．さらに，両者を組み合わせることで，セキュリティの問題を早期に明らかにし，要求定義段階で顧客などのステークホルダへの詳細な説明を可能にすることで，現状では困難なセキュリティ要求についての合意を促し，2 章で述べた，顧客要件によりセキュリティの対処が見送られるような事態を回避できるようになることも期待される．

筆者らは本提案手法を用いて，少人数のセキュリティ有識者で構成されるチームによる Web アプリケーション SI のセキュリティ要求定義支援作業を開始し，提案手法の適用により，必要な作業量（セキュリティ有識者単独，共同作業ともに）が，従来の脅威分析に比べ少ない量ですむことを確認した．またその結果定義された要求はセキュリティ対策として要求レベルでは十分であることを確認できた．しかし適用結果では，要求としては満たしていても，不適切な設計仕様や実装により脆弱性が入りこんでしまった例もあった．本提案手法は，要求分析段階で，実装段階で起きる脆弱性（インジェクション脆弱性など）を予測し，対策を用意することも可能にしている．しかし，実装で起きる脆弱性をすべて防止できるわけではない．実装で起きる脆弱性を防止するためには，要求どおりに実装させるように開発を制御したり，要求に違反していないかどうかチェックする仕組みが別に必要になる．これは本提案手法の限界であり，今後検討すべき課題である．

さらに，本提案手法を Web アプリケーションのドメインに適用した場合，いくつかの典型的な場合については同様の脅威や対策が抽出されることが分かった．脅威，対策抽出を類型化することにより，さらに作業効率を上げることが期待できる．この詳細検討については今後の課題である．

参 考 文 献

1) 情報処理推進機構：情報セキュリティ白書 2008，実教出版 (2008).
2) F-SECURE: Mass SQL Injection (online) (2008). http://www.f-secure.com/weblog/archives/00001427.html (accessed 2008-06-28)
3) Sindre, G. and Opdahl, A.L.I.: Eliciting Security Requirements by Misuse Cases, *Proc. TOOLS Pacific 2000*, pp.120–131 (2000).
4) 情報処理推進機構：セキュア・プログラミング講座 (online). http://www.ipa.go.jp/security/awareness/vendor/programmingv2/ (accessed 2008-06-28)
5) CERT: CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests (online) (2000). http://www.cert.org/advisories/CA-2000-02.html (accessed 2008-06-28)
6) WASC: Web Application Security Consortium Threat Classification: SQL Injection (online) (2007). http://www.webappsec.org/projects/threat/classes/sql_injection.shtml (accessed 2008-06-28)
7) OMG: Unified Modeling Language (online) (2007). http://www.uml.org/ (accessed 2008-06-28)
8) Haley, C.B., Moffett, J.D., Laney, R. and Nuseibeh, B.: A framework for security

requirements engineering, *2006 International Workshop on Software Engineering for Secure Systems* (*SESS '06*) *in Conjunction with the 28th International Conference on Software Engineering* (*ICSE*) *2006*, Shanghai, China, pp.35–42 (2006).
9) Haley, C.B., Moffett, J.D., Laney, R. and Nuseibeh, B.: Security Requirements Engineering: A Framework for Representation and Analysis, *IEEE Trans. Softw. Eng.*, Vol.34, No.1, pp.133–153 (2008).
10) Howard, M. and Lipner, S.: *The Security Development Lifecycle*, Microsoft (2006).
11) CC: Common Criteria for Information Technology Security Evaluation v3.1 (online) (2007). http://www.commoncriteriaportal.org/public/developer/index.php?menu=2 (accessed 2008-06-28)
12) Mead, N.R., Hough, E.D. and StehneyII, T.R.: Security Quality Requirements Engineering (SQUARE) Methodology (2005).
13) Howard, M. and LeBlanc, D.: *Writing Secure Code Second Edition*, Microsoft (2003).
14) Swiderski, F. and Snyder, W.: *Threat Modeling*, Microsoft Press (2004).
15) van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour.
16) Yu, E.S.-K.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering, *Proc. 3rd IEEE Int. Symp. on Requirements Engineering* (*RE'97*) (1997).
17) Bresciani, P., Giorgini, P. and Mouratidis, H.: On Security Requirements Analysis for Multi-Agent Systems, *2nd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems SELMAS 2003 in Conjunction with the 25th International Conference on Software Engineering* (*ICSE 2003*), Portland, Oregon, USA (2003).
18) Gani, A., Giorgini, P., Manson, G. and Mouratidis, H.: Analysing Security Requirements of Information Systems Using Tropos, *The 5th International Conference on Enterprise Information Systems*, Angers, France (2003).
19) van Lamsweede, A.: Elaborating Security Requirements by Construction of Intentional Anti-Models, *The 26th International Conference on Software Engineering*, pp.148–157 (2004).
20) Jürjens, J.: *Secure Systems Development with UML*, Springer-Verlag (2004).
21) Fernandez, E.B., VanHilst, M., Petrie, M.M.L. and Huang, S.: Defining Security Requirements through Misuse Actions, *Advanced Software Engineering: Expanding the Frontiers of Software Technology*, Ochoa, S.F. and Roman, G.-C. (Eds.), International Federation for Information Processing, pp.123–317, Springer (2006).
22) Braz, F., Fernandez, E.B. and VanHilst, M.: Eliciting security requirements through misuse activities, *accepted for the 2nd Int. Workshop on Secure Systems Methodologies using Patterns* (*SPattern'07*), *In Conjunction with the 4th International Conference onTrust, Privacy & Security in Digital Busines* (*TrustBus'07*), Turin, Italy (2008).
23) Jackson, M.: *Problem Frames: Analyzing and Structuring Software Development Problems*, Addison-Wesley (2001).
24) Haley, C.B., Laney, R.C. and Nuseibeh, B.: Deriving Security Requirements from Crosscutting Threat Descriptions, *3rd International Conference on Aspect-oriented Software Development*, pp.112–121 (2004).
25) ISO/IEC: ISO/IEC TR 15446, Guide for the Production of PPs and STs, Version0.93 2002-10-20 (online) (2002). http://www.ipa.go.jp/security/ccj/documents/27n3374ppstguide_v093.pdf (accessed 2008-06-28)

(平成 21 年 1 月 28 日受付)
(平成 21 年 7 月 2 日採録)

大久保隆夫（正会員）

1966 年生．1991 年東京工業大学大学院総合理工学研究科物理情報工学専攻修了．同年株式会社富士通研究所入社．リバースエンジニアリング，コンカレントエンジニアリング，モデル駆動開発の研究，開発に従事．現在は，システム・インテグレーション開発を中心としたアプリケーション開発のセキュリティ支援を担当．2009 年情報セキュリティ大学院大学博士課程修了，情報学博士．アプリケーションセキュリティ，Web セキュリティ，ソフトウェア工学に興味を持つ．

田中　英彦（名誉会員）

1943 年生．1965 年東京大学工学部電子工学科卒業．1970 年同大学院博士課程修了，工学博士．1987 年東京大学工学部教授，2001 年同情報理工学系研究科教授・研究科長，平成 15 年情報セキュリティ大学院大学教授・研究科長．計算機アーキテクチャ，人工知能，分散処理，メディア処理，デペンダブルシステム，セキュアシステム等に興味を持つ．著書に『非ノイマンコンピュータ』『計算機アーキテクチャ』『VLSI コンピュータ』『Parallel Inference Engine』等．本会名誉会員．IEEE フェロー．電子情報通信学会，ソフトウエア化学会，人工知能学会，ACM 各会員．