(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-533075
(P2004-533075A)
(43)公表日　平成16年10月28日(2004.10.28)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 15/00	G06F	15/00 330A	5B085

審査請求 未請求　予備審査請求 有　（全 77 頁）

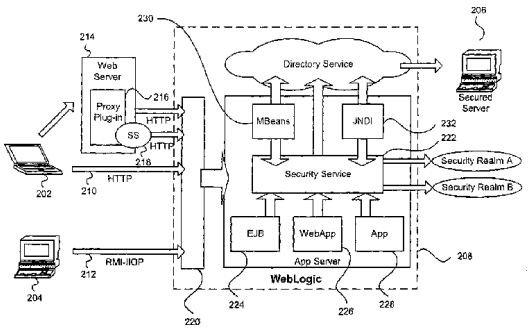
(21)出願番号	特願2003-504584(P2003-504584)
(86)(22)出願日	平成14年5月29日(2002.5.29)
(85)翻訳文提出日	平成15年12月11日(2003.12.11)
(86)国際出願番号	PCT/US2002/016644
(87)国際公開番号	WO2002/101973
(87)国際公開日	平成14年12月19日(2002.12.19)
(31)優先権主張番号	09/878,536
(32)優先日	平成13年6月11日(2001.6.11)
(33)優先権主張国	米国(US)

(71)出願人　500105160
ビーイーエイ　システムズ，　インコーポレイテッド
ＢＥＡ　Ｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国　カリフォルニア　９５１３１，　サン　ノゼ，　ノース　ファースト　ストリート　２３１５
２３１５　Ｎｏｒｔｈ　Ｆｉｒｓｔ　Ｓｔｒｅｅｔ，　Ｓａｎ　Ｊｏｓｅ，　ＣＡＬＩＦＯＲＮＩＡ　９５１３１　Ｕ．Ｓ．Ａ．
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100067013
弁理士　大塚　文昭

最終頁に続く

(54)【発明の名称】サーバーセキュリティ及び権限付与処理のためのシステム及びその方法

(57)【要約】
差し込み可能なアーキテクチャは、セキュリティ及びビジネスロジックのプラグインが、サーバー(206)をホストとするセキュリティサービスの中に挿入され、かつ、そのサーバー上、セキュリティドメイン(208)内の他のサーバー上、又はセキュリティドメイン間に存在する一以上の安全なリソースへのアクセスを制御することを可能にする。セキュリティサービス(222)は、セキュリティ実行、及びアクセス権判定における焦点として動作し、及び、一つのログイン処理内で使用又は判定された情報は、透過的かつ自動的に他のログイン処理に流通させることができる。権限付与は、特定のユーザー(204)が、特定の状況の中で、特定のリソース(226)を用いて、出来ること又は出来ないことを示している。権限の付与は、安全性の確保された環境の技術的側面(許可又は拒否の概念)を反映するだけでなく、サーバー提供者が要求するビジネスロジック又は機能を表現するためにも使用される。このようにして、権限の付与は、単純なセキュリティプラットフォームと複雑なビジネスポリシーのプラットフォームとの間の隔たりを埋める。



【特許請求の範囲】
【請求項１】
クライアントが保護されたリソースにアクセスすることを可能にするセキュリティシステムであって、
保護されたリソースへのアクセスについての、クライアントアプリケーションからのアクセス要求を受け取り、前記アクセス要求をセキュリティサービスに伝えるアプリケーションインターフェース機構と、
前記アクセス要求を許可又は拒否する判定を行うセキュリティサービスと、
許可されたアクセス要求を前記保護されたリソースへ伝えるリソースインターフェースと、
を備えるセキュリティシステム。
【請求項２】
前記アプリケーションインターフェース機構が、アプリケーション配置記述を読み取り、前記配置記述をセキュリティサービス内に登録するためのアプリケーションコンテナを含む、
請求項１記載のセキュリティシステム。
【請求項３】
前記アプリケーションコンテナが、EnterpriseJavaBeansコンテナである、
請求項２記載のセキュリティシステム。
【請求項４】
前記アプリケーションコンテナが、WebAppコンテナである、
請求項２記載のセキュリティシステム。
【請求項５】
前記セキュリティサービスが、アクセスポリシーを定義し、前記アクセス要求を許可、拒否、又は保留の寄与判定を定めるための複数のアクセス判定機構を含む、
請求項１記載のセキュリティシステム。
【請求項６】
前記セキュリティサービスが、前記アクセス要求を前記複数のアクセス判定機構に転送し、前記寄与判定を合体させて、前記アクセス要求を許可又は拒否するセキュリティサービスによる一つの全体判定とするアクセス制御装置をさらに含む、
請求項５記載のセキュリティシステム。
【請求項７】
前記アクセス判定が、ビジネス機能に関連するアクセスポリシーを表すものである、
請求項５記載のセキュリティシステム。
【請求項８】
アクセスポリシーにおける変更を反映するために、アクセス判定がセキュリティサービスに付加される、
請求項５記載のセキュリティシステム。
【請求項９】
前記保護されたリソースに前記クライアントがアクセスすることができる権限の付与を定義するために、前記アクセス判定機構が使用される、
請求項５記載のセキュリティシステム。
【請求項１０】
前記アクセス判定機構のいずれか一つによる拒否又は保留によって、セキュリティサービスがアクセス要求を拒否するようになった、
請求項５記載のセキュリティシステム。
【請求項１１】
前記アクセス判定機構のいずれか一つによる保留によっては、セキュリティサービスがアクセス要求を拒否しないようになった、
請求項５記載のセキュリティシステム。

【請求項１２】
前記セキュリティサービスが、前記複数のアクセス要求に対する判定を検査するための検査メカニズムをさらに含む、
請求項５記載のセキュリティシステム。
【請求項１３】
前記リソースインターフェースが、保護されたリソースへの要求、又は保護されたリソースからの要求を通すためのインターフェース機構を含む、
請求項１記載のセキュリティシステム。
【請求項１４】
前記インターフェース機構が、JavaJ2EEセキュリティインターフェースを含む、
請求項１３記載のセキュリティシステム。
【請求項１５】
前記インターフェース機構が、セキュリティプロバイダーインターフェースを含む、
請求項１３記載のセキュリティシステム。
【請求項１６】
前記インターフェース機構が、前記リソースインターフェースの中にプラグインとして含まれる、
請求項１３記載のセキュリティシステム。
【請求項１７】
セキュリティサービスがさらに、前記保護されたリソースから前記クライアントへの、前記アクセス要求に対する応答を許可するか又は拒否するかを判定する、
請求項１記載のセキュリティシステム。
【請求項１８】
クライアントが保護されたリソースにアクセスすることを可能にする方法であって、
保護されたリソースへのアクセスについてのクライアントアプリケーションからのアクセス要求を、アプリケーションインターフェース機構で受け取り、前記アクセス要求をセキュリティサービスに伝えるステップと、
前記セキュリティサービスにおいて、前記アクセス要求を許可するか、又は拒否するかを判定するステップと、
許可されたアクセス要求を、リソースインターフェースを介して前記保護されたリソースへ伝達するステップと、
を含む方法。
【請求項１９】
前記アプリケーションインターフェース機構が、アプリケーション配置記述を読み取り、
前記配置記述をセキュリティサービス内に登録するアプリケーションコンテナを含む、
請求項１８記載の方法。
【請求項２０】
前記アプリケーションコンテナがEnterpriseJavaBeansコンテナである、
請求項１９記載の方法。
【請求項２１】
前記アプリケーションコンテナがWebAppコンテナである、
請求項１９記載の方法。
【請求項２２】
複数のアクセス判定機構を介して前記セキュリティサービス内にアクセスポリシーを定義するステップと、
各アクセス判定機構において、前記アクセス要求を許可、拒否、又は保留するための寄与判定を定めるステップと、
をさらに含む、請求項１８記載の方法。
【請求項２３】
前記アクセス要求を、アクセス制御装置を介して、前記複数のアクセス判定機構に転送し

、前記アクセス要求を許可又は拒否するために、前記寄与判定を合体させてセキュリティサービスによる一つの全体判定とするステップ、
をさらに含む請求項２２記載の方法。
【請求項２４】
前記アクセス判定が、ビジネス機能に関連するアクセスポリシーを表すものである、
請求項２２記載の方法。
【請求項２５】
アクセスポリシーにおける変更を反映するために、アクセス判定がセキュリティサービスに付加される、
請求項２２記載の方法。
【請求項２６】
前記保護されたリソースに前記クライアントがアクセスすることができる権限の付与を定義するために、前記アクセス判定機構を使用するステップ、
をさらに含む、請求項２２記載の方法。
【請求項２７】
前記アクセス判定機構のいずれか一つの拒否又は保留によりセキュリティサービスがアクセス要求を拒否するようになった、
請求項２２記載の方法。
【請求項２８】
前記アクセス判定機構のいずれか一つの保留によってはセキュリティサービスがアクセス要求を拒否しないようになった、
請求項２２記載の方法。
【請求項２９】
検査機構を介して前記複数のアクセス要求の判定を検査するステップ、
をさらに備える、請求項２２記載の方法。
【請求項３０】
リソースインターフェースを介して伝達する前記ステップが、インターフェース機構を介して、保護されたリソースへの、あるいは保護されたリソースからの要求を通すステップを含む、
請求項１８記載の方法。
【請求項３１】
前記インターフェース機構がJavaJ2EEセキュリティインターフェースを含む、
請求項３０記載の方法。
【請求項３２】
前記インターフェース機構がセキュリティプロバイダーインターフェースを含む、
請求項３０記載の方法。
【請求項３３】
前記インターフェース機構が、前記リソースインターフェースの中でプラグインとして含まれる、
請求項３０記載の方法。
【請求項３４】
前記保護されたリソースから前記クライアントへの前記アクセス要求に対する応答を、許可するか、又はは拒否するかの判定を行うステップ、
をさらに備える、請求項１８記載の方法。
【請求項３５】
安全性環境における保護されたリソースへのユーザーのアクセス権限の付与を定める方法であって、
保護されたリソースへのアクセスについてのユーザーアプリケーションからのアクセス要求を受け取るステップと、
前記アクセス要求で、セキュリティサービスを呼び出すステップと、

前記保護されたリソースへのアクセスについてのユーザーの権限付与を定めるステップと、
前記セキュリティサービスにおいて、前記ユーザーへの権限付与に基づき、前記アクセス要求を許可、又は拒否する判定を行うステップと、
(a)許可されたアクセス要求を前記保護されたリソースへ伝えるステップ、又は(b)拒否されたアクセス要求を前記保護されたリソースに与えないステップ、のいずれか一方のステップと、
を含む方法。
【請求項３６】
前記権限付与が、前記保護されたリソースのユーザーに利用可能なアクセスの形式を定めるものである、
請求項３５記載の方法。
【請求項３７】
前記アクセスの形式が、前記保護されたリソースの一部又は全部を、開く、修正する、削除する、コピーする、のいずれかを含む、
請求項３６記載の方法。
【請求項３８】
前記ユーザーの権限付与についての情報を、第一のセキュリティ領域から第二のセキュリティ領域へ伝えることができる、
請求項３５記載の方法。
【請求項３９】
前記ユーザーの権限付与についての情報を、前記第一のセキュリティ領域から前記第二のセキュリティ領域へ伝える前に、前記第一のセキュリティ領域からの付加的情報を使用してユーザーの権限付与を修正することができる、
請求項３８記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般的にサーバーセキュリティ機構に関するものであり、特にサーバーセキュリティのためのアーキテクチャに関するものである。
【背景技術】
【０００２】
サーバー技術、特定的にはeコマースのサーバーに適用される「セキュリティ」という用語は、この数年の間に、セキュリティ環境のいくつかの側面をカバーするまでに拡張されてきた。この用語の元来の定義は、特定のユーザー(あるいはクライアント)を認証したり、またアクセスを許可されていないサーバー上の情報を、ユーザーが検索、閲覧、あるいはアクセスしないようにする機構を、常に想定していた。
【０００３】
図１は、今日一般に使用されているセキュリティ機構の典型例を示している。クライアント102,104は、多数のプロトコル、例えばhttp110及びII0P112を介して、アプリケーションサーバー106のようなセキュリティの確保されたリソースにアクセスする。アクセス要求は、典型的には物理的に固定された一連の規則に基づいて、要求されたアクセスを許可するか又は許可しないかを判定するセキュリティレイヤー118を通して、フィルターされる。セキュリティレイヤーは、保護されたリソース(たとえば、アプリケーションサーバーそのもの)のコンポーネントとすることができ、又は別の実体(たとえば、ファイアウォールシステムやデバイスの一部)として動作するものとすることもできる。
【０００４】
現在のシステムでは、アクセス要求の性質、クライアントの形式、又は保護されたリソースの形式を解析する試みは、殆ど又は全くなされていない。従って、セキュリティは、サーバーによって提供される保護されたリソースを理解し、それらのリソースへのユーザー

やクライアントのアクセス権を支配する、あらかじめ定義された一連の規則を理解するような形態に設計された単純な「許可又は拒否」機構として考えられることが多い。要求が作成される方法、又は特定のユーザーがリソースへのアクセスを要求きるようにする環境設定についての理解は、殆ど又は全くなされず、またセキュリティ機構は、セキュリティについてのビジネスポリシーにおける新たな変更を反映するための規則の容易な修正には適していない。

【０００５】

セキュリティの概念、とりわけユーザー特有の環境を表している情報を含むサーバーセキュリティの概念、及び特定の安全性の確保されたリソースへのアクセス要求の表現方法を発展させる試みがなされてきた。同様に、ビジネスポリシー、セキュリティポリシー、及びアクセス権における変更を反映するために、容易に修正可能なセキュリティの仕組みを提供する試みもなされている。一般的に、これらの方法は、一連のビジネスポリシーについての質問を一緒に集め、「これはＯＫですか？---このアクセスはセキュリティ規則に従って許可されますか？」と問いかけることを、依然としてアプリケーションプログラマーの責任とする。この過程を全て含んでおり、セキュリティ関連ビジネスポリシーの質問を尋ねるプログラム及びアプリケーションを開発できるようにするのに、プログラマーがビジネスの意味及びビジネスポリシーを理解する必要がないようにする機構が必要とされている。

【０００６】

最近では、リソースの安全性確保に関する新しい一連の要求が、アプリケーションサーバーの顧客とシステムインテグレーターとの話し合いから生じている。これらの新たな要求は、構造基盤の代わりとしてのアプリケーションという観点からのセキュリティとして表現される。これらの新たな要求の中での主要な区別は、ビジネスポリシー実行の存在である。ビジネスポリシーの実行は、典型的には、ユーザーが行為する役割及びビジネス要求の状況に基づいて、ユーザーが遂行することの「権限が与えられた」行為として記述される。顧客及びシステムインテグレーターは、アプリケーションの中にビジネスポリシーを実行するコードを埋め込まなければならないことに、失望している。この形式のロジックの埋め込みは、ビジネスポリシーが変更される度に、アプリケーションが修正され、テストされ、かつ再配置されなければならないという点で、配置問題を生み出す。ビジネスポリシーの変更のレートが与えられると、修正及び再配置に対する現在の要求は受け入れられないものとなる。

【０００７】

顧客及びシステムインテグレーターは、理想的にはこれらの新しい承認能力を、異なる形式の実行及びリソースコンテナ、例えばEnterpriseJavaBean(EJB)、WebApplications(Servlet、JSP)のためのものを含む実行及びリソースコンテナ、並びに他の形式のビジネス及びリソースコンテナにわたって、普遍的に適用できるようにしたいのであろう。役割ベースのアクセス制御(RBAC)は、顧客及びシステムインテグレーターが求める主要な許可形式の一つとなりつつある。役割を使用する傾向は、承認判定を行う時に、組織の識別情報を識別の抽象的な形態として用いることを許す。役割の使用は、管理全体を単純化するため、アプリケーションセキュリティの管理におけるコスト及びエラーの両方を軽減する機構を提供する。

【０００８】

Java2EnterpriseEdition(J2EE)は、役割ベースのアクセス制御の宣言形態を定義する。しかしながら、それは宣言計画に基づくため、主体に対する役割の関連付けは静的なものである。さらに、現在のJ2EEの仕様は、与えられた主体と関連付けられる役割を決定する際に、要求のパラメータ又はターゲットの識別のようなビジネス要求の状況を考慮に入れることができる手段を、全く提供していない。その結果、アプリケーション開発者は、「オーナー」のような概念をサポートするために、動的な役割の関連付けを計算するビジネスポリシーの規則を、アプリケーション内に実装することが求められる。

【０００９】

近頃では、顧客及びシステムインテグレーターは、Java2セキュリティで定義されるパーミッションベースのアクセス制御や、Java2EnterpriseEditionで定義される役割ベースのアクセス制御によって提供されるもの以上に、より豊かな一連の承認能力を要求している。より高レベルの承認に対する要求の基本は、アクセス制御リスト及びビジネスポリシー実行といった、従来の承認機構の共通部分に見ることができ、また承認判定を行う際に、ビジネス要求の状況を考慮に入れることができる能力を含むこともできる。要求の状況は、ターゲットオブジェクトの識別情報、要求のパラメータの値、及び手続開始クライアントのネットワークアドレス又はIPアドレスのような潜在的な環境情報を含んでいる。

【００１０】

実行又はリソースコンテナの形式に関係なく、これらの新しい承認能力の統合のための単純な機構の欠如が、顧客及びシステムインテグレーターの失望の原因となっている。サービスプロバイダーインターフェース(SPI)は、外部の承認プロバイダーとの統合を許すための機構であり、カリフォルニア州サンノゼのBEAシステムIncの製品であるWebLogicサーバー等のいくつかのアプリケーションサーバーによって用いられている。このSPIの領域は、第三者承認機構を統合するための有効な手段として用いられる能力を制限するか、あるいは顧客及びシステムインテグレーターに新しい承認機能が求めらられるという点で、多くの限定がある。

【００１１】

現在の「領域」SPIの最も大きな制限の一つは、実行の範囲である。現在、領域SPIの機構の実行範囲は、EnterpriseJavaBean及びWebアプリケーションのようなリソースをカバーしない。特に領域SPIは、JMSの宛先、JNDIのエントリー、及びきめの粗いレベルのサーブレットのようなRMI形式のリソースに焦点が当てられる。そのようなリソースの保護をサポートするのに必要な承認ポリシーの定義を維持するために、領域SPIはアップデートできるのであるが、これが、最終的に領域SPIをすべての承認要求を受け入れるのには非現実的な機構にしてしまう他の制限である。

【００１２】

現在の領域SPIの第二の制限は、実行のポイントである。現在の領域SPIの機構は、保護されたリソースへのアクセスが領域自身の中に存在することを許すような判定がなされるポイントを許さない。そのかわり、実行のポイントは、アプリケーションサーバー自身の中にある。このアプローチのため、現在の領域SPIは、パーミッションベースの承認機構のみをサポートするように定義されており、領域SPIは単にアクセス制御リストのデータベースとして動作する。同様に、Java2sandboxもサポートできるJava2のポリシーオブジェクトを提供するほどの複雑さは、殆どのベンダーで受け入れられていない。

【００１３】

さらに他の制限は、現在の領域SPIの実行機構のサポートである。実行ポイントについては、現在の領域SPIによって許されている実行の機構は、パーミッションベースの承認機構に基づくもののみに限定される。これが、第三者承認プロバイダーを導く能力に反するものであり、これらプロバイダーの統合は、日常ベースで顧客及びシステムインテグレーターによって望まれている。

【００１４】

これらの制限は合わさって、プロバイダーの価値提供を最小化することなくエンタープライズアプリケーションサーバーと統合することができる承認プロバイダーの形式を抑制する。求められている高度の承認を提供するために、要求の状況を獲得するような能力は現在残っていない。

【００１５】

(優先権の主張)

本出願は、2001年6月11日に出願された、「サーバーセキュリティ及び権限付与処理のためのシステム及びその方法」という名称の米国特許出願第09/878,536号に基づく優先権を主張するものであり、この出願を引用により本出願の中に組み入れる。

【００１６】

(発明の概要)
本発明は、一般的に、サーバーセキュリティ、特にサーバーセキュリティと権限付与処理を提供するアーキテクチャに関するものである。差し込み可能なアーキテクチャは、セキュリティ及びビジネスロジックのプラグインが、サーバーをホストとするセキュリティサービスの中に挿入され、かつ、そのサーバー上、セキュリティドメイン内の他のサーバー上、又はセキュリティドメイン間にある一以上の安全性の確保されたリソースへのアクセスを制御することを可能にする。セキュリティサービスは、セキュリティ実行、及びアクセス権判定における焦点として動作し、また、ある一つのログインプロセスの中で使用又は判定された情報は、他のログイン処理にも透過的かつ自動的に流通させることができる。

【００１７】
本発明はまた、アクセス状況内で使用される権限付与の概念を導入する。本出願の文脈の中で使用される、「ユーザー」又は「クライアント」は、同一の事－実際の人間、実際の人間の管理下にあるハードウェアデバイス或いはソフトウェアアプリケーション、又はユーザー介入なしの自己管理下で動作するハードウェアデバイス或いはソフトウェアアプリケーションのいずれか、をいう。ユーザー（又はクライアント）は、サーバー上にある保護されたリソースへアクセスしようとする実体である。この保護されたソフトウェアとは、たとえばサーバー上で動作するソフトウェアアプリケーション、特定のウェブページ或いはウェブサイトの一部、又はデータベース等である。

【００１８】
ユーザーがリソースにアクセスしようとする時、セキュリティサービスは、アクセス要求の形式、宛先(保護されたリソース)、及びその要求が作成された環境－以下、アクセス状況または単に状況と呼ぶ、を判定する。この情報から、セキュリティサービスは、ユーザーに対する「権限の付与」、又は一連の権限の付与を決定することができる。権限の付与は、特定のユーザーが、特定の状況で、特定のリソースを用いて、出来ること又は出来ないことを明白に示すものである。権限の付与は、安全性の確保された環境の技術的側面(許可又は拒否の概念)を反映するのみならず、サーバー提供者が要求するビジネスロジック、又は機能を表すことにも使用される。このようにして、権限の付与は、単純なセキュリティプラットフォームと、複雑なビジネスポリシープラットフォームとの間の隔たりを埋める。

【００１９】
その能力を説明するため、次のようなビジネス例を考える：
「スミス医師は、患者の医療カルテを更新することができますか？」という質問に対する答えは、その質問が尋ねられた状況に依存する。パーミッションベースの承認システムでは、リソースは『医療カルテ』オブジェクトのある事例であり、要求は『更新する』ことであり、そして対象は『スミス医師』であるので、状況はない。従って、もし返された答えが『はい』ならば、スミス医師はいかなる患者のカルテでも更新することができる。能力ベースの承認システムでは、質問中に、どの患者の、という必要な状況を付加することが可能である。従って、この質問は、「スミス医師は、ジョン・ジョー氏の医療カルテを更新することができますか？」というように言い換えられることができる。この質問に対する回答を判定するにあたり、スミス医師がジョン・ジョー氏の個人的な医者であるのか、又はおそらく医療センターでの担当医であるのか、を知る必要がある。単純な循環を使って、その概念を次のように表すことができる。

【００２０】
ある実施形態では、本発明は、クライアントが保護されたリソースにアクセスすることを許し、保護されたリソースへのアクセスについてのクライアントアプリケーションからのアクセス要求を受け取り、及び前記アクセス要求をセキュリティサービスに伝えるアプリケーションインターフェース機構と；前記アクセス要求を許可または拒否する判定を行うセキュリティサービスと；許可されたアクセス要求を前記保護されたリソースに伝えるリソースインターフェースとを備えるセキュリティシステム、を備えるものである。

【００２１】
他の実施形態では、本発明は、クライアントが保護されたリソースにアクセスすることを許し、保護されたリソースへのアクセスについてのクライアントアプリケーションからのアクセス要求をアプリケーションインターフェース機構で受け取り、前記アクセス要求をセキュリティサービスに伝えるステップと；前記セキュリティサービスで、前記アクセス要求を許可又は拒否する判定を行うステップと；リソースインターフェースを介して、許可されたアクセス要求を前記保護されたリソースへ伝えるステップとを備える方法、を備えるものである。
【００２２】
(発明の詳細な説明)
本発明の実施形態は、サーバーセキュリティ及び権限付与処理を提供し、セキュリティ及びビジネスロジックプラグインがサーバーをホストとするセキュリティサービスに挿入されることを許し、かつ、そのサーバー上、そのセキュリティドメイン或いはセキュリティ領域内の別のサーバー上、又はセキュリティ領域間に存在する一以上の安全性の確保されたリソースへのアクセスを制御するのに使用されるセキュリティアーキテクチャを含む。セキュリティサービスは、セキュリティ実行及びアクセス権判定における焦点として動作し、かつ、シングルサイン又はセキュリティ実行を考慮して、一つのログイン処理内で使用された情報は、自動的に他のログイン処理にも流通させることができる。
【００２３】
以下で定義される新しい用語を除いて、本文書で使用される用語は、Java、EnterpriseJavaBeans、WebLogicサーバーに関する標準書で定義される専門用語、及びその他一般的に認められているセキュリティ概念と一致するものである。
アクセス制御 - 承認されないリソースの使用を防ぐための、リソースへのアクセス制限
アクセス制御情報(ACI) - リソースへのアクセス要求の発起者についての情報であり、アクセス制御実行の判定を行うために使用される。
アクセス制御リスト(ACL) - リソースにアクセスすることを承認された実体を、それらのアクセス権と共に示すリスト。
承認 - 権限の許可であり、アクセス権に基づくアクセスの許可を含む。
資格 - ユーザー又はその他の主体のセキュリティ属性(識別情報及び/又は特権)を記述する情報。資格は、認証又は権限の委任を通して主張され、かつアクセス制御によって使用される。
権限の委任 - おそらく制限を伴って、一人のユーザーあるいは主体が、他者に、自分の識別情報又は特権を使用する権限を与える行為。
権限の付与 - ある主体に認められた権利または許可。
識別情報 - 一意性を持つセキュリティ属性であり、如何なる二つの主体の識別情報も同一であることはない。主体は、それぞれ一意の、いくつかの異なった種類の識別情報を有する。
主体 - システムのリソースを使用できる能力を持つユーザー、又はあらかじめプログラムされた実体。
役割 - 承認されたユーザーに対する一連の許容される行為を定義する、組織的な識別情報。
役割ベースのアクセス制御(RBAC) - 役割と呼ばれる組織的な識別情報を通じて、リソースへのアクセスを調停するセキュリティ機構のクラス。
セキュリティ属性 - その対象を支配するシステムポリシーの基礎を形成する、対象(ユーザーまたは主体)の特性。
セキュリティポリシー - システムのセキュリティサービスが何の保護を提供しなければならないか、を定義するデータ。アクセス制御ポリシー、検査ポリシー等を含む多数の種類のセキュリティポリシーがある。
サービスプロバイダーインターフェース(SPI) - サービスの一部の具体的な実装をサポートするパッケージ、又は一連のパッケージ。

ターゲット - 権限委任における「呼び出し連鎖」中の、最終受信者。呼び出しの発信者ではない、呼び出し連鎖における唯一の参加者である。
ターゲットオブジェクト - ビジネス要求メッセージの受信者。
信託モデル - もしシステムを安全のまま保つのであれば、システム中のどのコンポーネント及びシステム外のどの実体を信託すべきか、及び何についてそれらを信託すべきか、についての記述。
信託演算処理基盤(TCB) - システムを安全に保持するために、正しく機能すべきシステムの部分。TCBは、耐不正性であることがのぞましく、そのポリシーの実行は妨げられてはならない。
認証されない主体 - 如何なる識別情報又は特権も認証されなかったユーザー、または他の主体。
ユーザー - 自分の代理としてオブジェクトにシステム内で機能を実行させるために、当該オブジェクトに要求を出すようにシステムを利用する人間。

【００２４】
ここで記述される本発明の実施形態は、アクセスに関連して使用される権限付与の判定を可能にしている。典型的には、ユーザーは、サーバー上の保護されたリソースにアクセスしようとする。この保護されたリソースとは、例えば、サーバー上で動作するソフトウェアアプリケーション、特定のウェブページ又はウェブサイトの一部、或いはデータベースシステムである。他の形式の保護されたリソースも、本発明の技術的範囲内である限り使用することができる。ユーザーがリソースにアクセスしようとする時、セキュリティサービスは、アクセス要求の形式、宛先(保護されたリソース)、及び要求が作成された状況を判定する。この情報から、セキュリティサービスは、ユーザーに対する、権限の付与、又は一連の権限付与を判定することができる。権限の付与は、特定のユーザーが特定の状況のもとで特定のリソースを用いて出来ることを、明白に示す。権限の付与は、サーバー提供者が要求するビジネスロジック又は機能を表すのに使用され、それゆえ、単純なセキュリティプラットフォームと複雑なビジネスポリシープラットフォームとの間の隔たりを埋めている。

【００２５】
本発明の実施形態は、さらに、以下のようなアプリケーション形式に保護を提供する第三者ベンダーのセキュリティ製品の統合も考慮に入れる：
・ EnterpeiseJavaBean
・ ウェブアプリケーション(Servlets、JavaServerPages(JSP's))
・ リソース（RemoteMethodInvocation(RMI)、JavaMessagingSystem(JMS))

【００２６】
本発明の実施形態は、また、以下のような高度な機能をサポートするため、外部公開鍵の構造基盤の統合も考慮に入れる：
・ 証明書と鍵の検索
・ 証明書の有効化
・ 無効化機構(証明書革新リスト(CRL)、オンライン証明書状態プロトコル(OCSP))

【００２７】
本発明を開発するにあたっての重要な目標は、Java2EnterpriseEdition(J2EE)の仕様、及びそれとの間の相互運用性のサポートを含むことである。このJ2EEの仕様の特徴は、共通安全相互運用性(CSI)プロトコル、ユーザー識別トークン、状態なし認証サービス(SAS)プロトコル、マシン、クラスタ、及び/又はドメインの境界間にわたるセキュリティ資格の伝搬のサポート、ポリシーに基づく識別情報の伝搬の制御、仮想ホスト/サイトの高度なサポート、ドメイン範囲内でのユーザー識別情報の生成能力、及びホスト/サイト特有のセキュリティポリシーを含む。

【００２８】
本発明の他の重要な目標は、更新されたSSL能力、ハードウェア加速器、トランスポート層セキュリティ(TLS)プロトコル、セッション再開、証明書認可局、ネットワーク接続の

保護、ウェブサーバーのプロキシ、サーバー対サーバー通信、セッションIDの暗号化、及びディレクトリサーバーとの高度な統合、のような機能のサポートを含むことである。

【００２９】

セキュリティドメイン

本発明の状況において使用される、セキュリティ領域又はセキュリティドメインは、一つのサーバー、複数のサーバー、一群のサーバー、又は管理ドメインに及ぶものである。本発明は、セキュリティドメイン内での、セキュリティドメインにわたる、及びセキュリティドメイン間における識別情報伝搬の管理制御を考慮している。クライアントアプリケーション又はユーザーの識別情報は、伝搬され、かつセキュリティドメインによって有効範囲を絞られる。たとえば：

Principal-Identity@security-domain-name

【００３０】

伝搬が動作不能の時は、手続開始識別情報は発信者不明(アノニマス)として表すことができる。たとえば：

⟨anonymous⟩@initiating-security-domain-name

【００３１】

フレームワーク

本発明によって提供されるセキュリティアーキテクチャ又はフレームワークは、出された要求及び多数の他の事に対処するよう意図されている。承認要求のみに焦点をあてると、このセキュリティアーキテクチャは、典型的なパーミッションベースの承認に加えて、次のような能力を提供しようと試みる。

【００３２】

動的な役割の関連付け

動的な役割の関連付けとは、役割の主体への関連付けの遅い結合を許す機構、すなわち、主体に権限付与されている機能を判定する間に、要求の状況を考慮に入れることができるようにする機構である。動的な役割の関連付けは、実行時における、主体の役割との遅い結合として考えることもできる。この遅い結合は、主体対役割の関連付けが静的に定義されるか、動的に計算されるかに関わらず、保護されたリソースに対する承認判定の直前に生ずる。呼び出しシーケンスにおける位置付けにより、主体対役割の関連付けについてのどんな結果も、この要求の一部としてなされた何れかの承認判定の一部としての識別情報として、受け取ることが出来る。

【００３３】

静的に定義された関連付けと異なり、主体の役割への関連付けは、実行時に動的に計算される。動的役割の計算は、ターゲットの識別情報(利用可能な場合)、要求のパラメータの値、手続開始主体に関連付けられたプロファイル属性の値、並びに潜在的な他の情報、を含む要求の背景を作り上げる多くの情報にアクセスすることができる。

【００３４】

状況の情報は、典型的には、規則又は式評価エンジンによって評価されることとなる式の中のパラメータ値として利用される。評価エンジンによって提供される機能に依存して、主体が飛んだ距離の大きさのような外部計算を行う外部クラスを呼び出す能力を表現することも可能であり、これはその時、式のパラメータとして使用される。

【００３５】

対象中の主体と動的に関連付けられるべき役割を計算することに加えて、この機能は、また、J2EEで定義された配置記述の使用を通じてか又は管理者によってか、のいずれかで静的に定義された役割を関連付ける任務もある。

【００３６】

この能力の結果は、一実施形態においては、対象に含まれる体が所定の状況及びターゲットを受け取る権限を付与された役割を含む更新されたJAAS Subjectである。これらの役割は、次いで、ビジネスコンテナ及びアプリケーションコードと同様に、保護されたリソー

ス上での何らかの承認判定に利用できる。たとえば、個人向けサーバー、アプリケーションサーバー、又はウェブサーバーは、特定の役割がウェブページを個人向けにする手段として対象によって保持されるか否か、を判定する標準的な方法を利用することが出来る。この機構は、アクセスが許可されるべきか否かを判定するビジネスポリシーの情報を持つことなく、フィールドレベルの承認という結果となる、データベースの記録からフィールドの検索を行うか否かを判定するために、EnterpriseJavaBean又は他のアプリケーションで使用される。

【００３７】

パラメータ承認

パラメータ承認とは、保護されたリソースについての承認判定が、潜在的に要求の状況に基づいてなされることを可能にする機構である。本発明によって提供されるアーキテクチャの範囲内で、その実施範囲は、全ての実行及びリソースコンテナに適用できるまでに拡張された。これは、すべての実行及びリソースコンテナに、セキュリティサービスによって提供されるアクセス制御装置オブジェクトを通して、承認及び役割のマッピングサービスを獲得させることにより、遂行される。このアクセス制御装置は、要求される能力を提供するため、サービスプロバイダーインターフェース(SPI)を通して提供される承認サービスを利用する。

【００３８】

動的な役割の関連付け及びパラメータ承認を提供するのに使用される機構を実際に定義するサービスプロバイダーインターフェースは、権限委任される承認デザインに基づくものであり、これは実行点がアプリケーションサーバー自身の中でなく機能の提供側に移るという結果を生ずる。

【００３９】

さらに、承認フレームワークを備えるサービスプロバイダーインターフェース上で定義される方法は、第三者承認ベンダーによって提供されるサービスとより自然に提携するように設計された実施機構を利用するものである。能力ベース機構の使用は、権限委任される承認手法が、能力ベースとパーミッションベースの両方の実装をサポートすることを可能にする。

【００４０】

従来のセキュリティ機構は、所定のリソースに対して主体に与えられた権限のみに基づくものであるので、状況なしの傾向が強い。それゆえ、なされる承認判定の唯一の形式は、主体がリソースにアクセスするのに必須な権限を持っているか否か、というものである。承認判定のこれらの形式は、ビジネス判定を表そうとするものではないので、より複雑になっている。結果として、この制限を補償するために、付加的なセキュリティチェックがアプリケーションコードの中でしばしば要求される。たとえば、従来の方法を使用することにより、主体が特定の口座にお金を転送することが許されるか否かについて、評価できた。しかしながら、送金の合計、発信元と送信先の両方の口座の通貨、及び曜日を考慮に入れることは不可能である。

【００４１】

本発明に従ったパラメータ承認ベースの機構では、要求の状況を使って承認判定がなされ、それゆえ、その承認判定は、より綿密に現実のビジネス判定を表している。アプリケーション自身の中で、セキュリティチェックを補う必要は殆どない。例えば、実社会の銀行業務の例に本発明を使用すると、誰によって、及びどのような環境下で送金が許されるか否かを管理するビジネスポリシーの処理の一部として、送金の合計、送信元と送信先の両方の口座の通貨、及び曜日を考慮に入れて、承認判定することが可能となる。

【００４２】

状況情報へのアクセス

いくつかの他のシステムが、上で述べたパラメータ承認の記述と同様な機能を提供しようとする一方で、これら先行のシステム及び方法はすべて、呼び出し元に対し、評価されているビジネスポリシーのパラメータについての以前の情報を持つように要求する。この情

報への要求は、アプリケーションコードービジネスポリシーのある側面の情報の中で承認判定を行う場合と、事実上、全く同じ問題を提示する。他の状況情報を要求するためにビジネスポリシーが変更される度に、アプリケーションは修正かつ再配置されなければならない。

【００４３】

ビジネスポリシーについての以前の情報なしで、状況情報を提供することの問題への対処は、承認プロバイダーからコンテナへの呼び戻しの使用によって遂行される。アプリケーションを根本的に変化させる要求のパラメータの形式又は数の変更が、アプリケーションの再配置を要求する一方で、動的な役割の関連付け及びパラメータ承認プロバイダーの実装が、アプリケーションに0ビジネスポリシー面の以前の情報を持つことを要求することなく、状況情報にアクセスすることが可能となる。

【００４４】

本発明を用いて動作し、かつ承認フレームワークをサポートするように定義されたサービスプロバイダーインターフェース(SPI)は、Java認証及び承認サービス(JAAS)で定義されるのと同様な、標準化された呼び戻し機構を利用することができる。コンテナからサービスプロバイダーインターフェースの実装へ渡される呼び戻し処理ルーチンを利用することで、プロバイダーは特定の状況情報が返されるよう要求できる。コンテナの呼び戻し処理ルーチンが呼ばれると、コンテナは、呼び戻しに適切な値を代入することにより応答する。もしコンテナが、処理ルーチンの中で指定される呼び戻しを理解しない場合には、コンテナは、呼び戻しに値の変更を反映しない。アプリケーションもコンテナもいずれも、特定のビジネスポリシーの式の評価によって導き出される状況情報を問われるものであるため、ビジネスポリシーに対する変更は意識しない。

【００４５】

機能説明

パラメータ承認の要求を満たすのに使用される、権限付与処理エンジン(権限付与エンジン)の実装は、次のような要求をサポートしなければならない：

【００４６】

実装は、ビジネス要求からのパラメータ値、主キー、認証されたユーザーを意味するJAAS対象内の主体からの属性、及び状況の一部と考えられ得るその他の何らかの情報が、規則式の一部として表現されることを明示できるべきである。

【００４７】

実装は、規則の評価によって使用される可能性のある如何なるパラメータ情報も、規則評価の最初に渡されることを要求するのではなく、呼び出し元からオンデマンドで、パラメータ情報を要求できるべきである。情報要求の機構は、JAASにおいて定義されるのと同様な呼び出し形式であるべきである。その実装は、個々の情報それぞれについて呼び戻しを行い、一つの呼び戻しの中で複数の情報を要求する、又はその組み合わせを行う。

【００４８】

実装は、規則評価の一部として呼び出されるJavaクラス名を指定できるべきである。Javaクラスから返される値は、規則式の評価の中で使用される。これは、規則を評価する際に考慮される値を計算するためにユーザー提供のコードが呼び出されるという、マクロ機能として考えられる。たとえば、旅行予約の例では、規則式は、対象によって表される乗客が現在までに飛行したマイル数を計算、又は調べるユーザー提供クラスへの呼び出しを指定することもできる。

【００４９】

一連の規則の評価は、規則が正常に満たされたか否かを示すブーリアン値となるべきである。

【００５０】

本発明は、即座に適応及びカスタマイズできるビジネス機能及びシナリオを可能にするために、配置又は「プラグイン」されるアプリケーションの実装、特にログイン、承認、及び検査実装の範囲内でのフレームワークアーキテクチャを、企業アプリケーション開発者

に提供する。

【００５１】

本発明によって提供されるフレームワークアーキテクチャは、開発者(又はユーザー)に見えない、かつセキュリティ処理を付加的に階層化又は変換する必要なく、提供されたセキュリティを企業全体に分散させることができる方法でのこれらセキュリティ機能の利用を大幅に拡張することを除いて、特に、Java開発者が既存のJavaセキュリティ機能を利用することを可能にしている。

【００５２】

本発明の実施形態は、特に、企業及びサーバーセキュリティ、及びビジネスワークフローの分野で役立つ。フレームワークは差し込み可能なアーキテクチャを使用するので、アーキテクチャの中核を修正する必要なく第三者ベンダーがセキュリティ実装のプラグインを提供することを可能にしている。これらのプラグインは、デジタル証明書及びキー検索、証明書の有効化及び無効化機構(例えばCRLやOCSP)のためのツールを含む。差し込み可能なフレームワークアーキテクチャは、それゆえ、EnterpriseJavaBeans(EJB's)、アプリケーション(Servlet及びJSP'sを含む)、及び他のネットワークリソース(例えばRMI,JNDI,JMSの宛先)、のような企業リソースの安全なアクセス制御を考慮に入れる。

【００５３】

さらに、本発明のフレームワークアーキテクチャは、特に、安全な承認、及び企業全体のセキュリティドメイン全域にわたる安全なリソースアクセスの方法を提供するのに適している。ドメインを持つ各サーバーは、異なる一連の機能、サービス、及びアプリケーションを提供する。本発明は、ドメイン全域にわたる又は協調ドメイン間でのクライアントのアクセス権限を判定する一回限定ログオンの機構を通して、各サーバーの特定のサービスを利用することができるクライアントアプリケーションを、開発者が配置することを可能にしている。

【００５４】

図２は、本発明の実施形態に従ったセキュリティアーキテクチャの例を示している。その中に示されるように、クライアント202,204(物理的なハードウェアクライアント、又はソフトウェアアプリケーション)は、トランザクション又はアプリケーションサーバー208を介して、持続的ディレクトリサーバーのような安全性の確保されたサービス又はリソース206にアクセスしようと試みる。そのようなトランザクションサーバーの一例として、カリフォルニア州サンノゼ所在のBEA Systems Inc.によるWeblogicサーバー製品があるが、本発明では、如何なる他のサーバー製品又は等価なシステムを使用することも可能である。インターネットCORBAのクライアントは、典型的には、InternetInter-ORBプロトコル(IIOP)要求212を通じて、そのようなアクセスを試みる。ウェブのクライアントは、典型的には、ウェブサーバー214を介して直接的に、又はプロキシプラグイン216(このケースでは、プロキシはまた、例えばSSL暗号化218のような付加的機能も提供する)を介してのいずれかにより、一連のhttp要求210を通してアクセスを試みる。いかなる場合においても、接続試行は、しばしば最初の接続フィルター220を介して、トランザクションサーバーに受け取られ、そしてセキュリティサービス222に渡される。本発明によれば、セキュリティサービス222は、クライアント及びユーザレベルのリソースアクセス、承認、証明、権限評価、及び権限付与の判定を含む、セキュリティ判定における焦点となる。EnterpriseJavaBeans(EJB's)224、ウェブアプリケーション(WebApp's)226、及び他の形式のアプリケーションはすべて、コンテナの使用を通じて、セキュリティサービスを利用する。図２の場合には、セキュリティサービスは、これらコンテナから保護されたリソースへの呼び出しを処理する。呼び出しは、例えば複数のMBeans230、又はJNDI232によって処理される。

【００５５】

図３は、セキュリティサービスのアーキテクチャ300の実施形態を、非常に詳細に示している。セキュリティサービスは、標準Java2EnterpriseEditionのセキュリティセットにより提供される基本的なセキュリティサービス及び機能を強化するものである。この例に示

されるように、基本的なJavaセキュリティセット302は、他との間の、鍵の保管、認証、証明書の有効化、及び安全なソケットのためのセキュリティプロバイダーインターフェース304を含む。顧客アプリケーション306は、Javaセキュリティ層、及びこれらSPI'sを直接利用するように書かれている。本発明は、図3に示すように、これら及び他のセキュリティ機能の利用における、開発者の選択の幅を大幅に広げている。本発明によれば、顧客アプリケーションは、例えばWebAppコンテナ310のEJBコンテナ308のようなコンテナの中に配置される。コンテナは直接、セキュリティサービス314(ここでは、セキュリティサービス222と同じ)と通信し、セキュリティサービス314は、Javaセキュリティ層302及びそのセキュリティSPI's304と、交替で通信する。これは、安全な承認判定に対する責務を、アプリケーションから移し、かつセキュリティサービス層に任せるとを可能にする。

【００５６】

さらに、本発明は、既存のJavaセキュリティSPI's304の使用を、特化されたSPI's316,318と統合するのに理想的である。例えば、接続フィルター、アクセス判定インターフェース、検査チャネル、及び証明書認証者のような付加的SPI'sは、セキュリティサービスに含まれる、又は統合される。セキュリティサービスは、安全な企業リソース全域へのアクセスを調停する。

【００５７】

図４は、アプリケーションが安全なリソースへアクセスするために本発明を使用する過程を示している。ステップ402で、開発者は顧客アプリケーションを作成する。アプリケーションは、実際のユーザーあるいは顧客によって使用されるものとすることができ、又は他のアプリケーションがセキュリティサーバーを利用できるようにするためのアプリケーションインターフェースプログラムとすることもできる。ステップ404で、アプリケーションはコンテナ内部に配置され、その配置過程については、以下でより詳細に開示する。次いで、アプリケーションは、ステップ406において、コンテナを用いてセキュリティサービスにアクセスする。任意で、ステップ408において、セキュリティサービスは、一連のJ2EE準拠のJavaセキュリティ機能と直接、整合性をとることができる。一方、ステップ410では、コンテナは、第三者又はカスタムのセキュリティプロバイダーインターフェースへのアクセスを可能にするのに使用される。ステップ412で、セキュリティプロバイダーインターフェースは、要求されたアクセスが許可されるべきであることをセキュリティサービスが示したのを受けて、アプリケーションが安全なリソースにアクセスすることを許可する。

【００５８】

図５は、コンテナ内部にアプリケーションを配置するのに使用される配置機構の、一実施形態を示している。図５に示すように、データは、アプリケーションコンテナ504による使用のため、セキュリティサービス502(ここでは、セキュリティサービス314と同じ)に登録される。従来的には、コンテナが配置記述506から情報を読み取り、それから実際のコンテナが実行する。もし、WebAppのコンテナ、EJBのコンテナというように、異なる種類のコンテナが存在する場合、異なる実装エンジンである必要がある。さらに、コンテナを持たず、かつ配置説明も持たないJ2EE環境の内部に、一連のリソースが存在する。この方法を用いる一番の問題は、セキュリティの様々な側面が、一つの環境の中で、異なった方法で実行されることである。本発明は、第一に、配置記述子を読み取ったコンテナが、その中で定義されているセキュリティ制限及びポリシーをセキュリティサービスに伝えることを可能にすることにより、この問題に対処するが、これはセキュリティサービス内部のセキュリティポリシーの集団となる。第二に、コンテナは、その後保護されたリソースへのアクセス要求を作成した時には、全ての承認判定をセキュリティサービスに委任し、従って、実行点が、コンテナからセキュリティサービスに移ることになる。配置記述を通して提供される、並びに管理ツールを通して管理者によって定義されるセキュリティ制限及びポリシーを参照することにより、承認判定がなされる。配置ツールは、また、配置記述子を読み取る、及びセキュリティサービス内部に存在するセキュリティ情報を記録するのにも使用される。配置ツールは、次からは、配置記述子を読み取ることなく、直接セキュ

リティサービスへ向かうことができる。

【００５９】

図６は、アプリケーションの配置の間に、データがセキュリティサービスの中に入り込む過程を示している。ステップ602で、開発者が、配置記述を作成する。その記述は、ステップ604でコンテナに渡され、コンテナは何の情報をセキュリティサービスに渡すべきか評価するため、その情報を解析する。任意で配置記述は配置ツールに与えられることもあり、ステップ606で、配置ツールはその情報をセキュリティサービスに渡す。

【００６０】

図７は、ネットワーク上又はサーバー上の保護されたリソースにアクセスするために、本発明に従ってセキュリティサーバーを使用するクライアントの一例を示している。その処理は、要求を作り出すクライアント702で始まり、これは、自分自身を認証するクライアントという単純な場合の一例である。クライアントは、矢印704で示されるように、コンテナに接続する。トークン706は、要求と共に渡される。接続確立の一部として、システム(セキュリティサーバー710を含む)は、リソース708が保護されているかを判定し、そうであれば、クライアントは認証する必要がある。この例では、トークンは、クライアントの認証されたユーザーを表しており、呼び出し要求の中で順番にコンテナへ渡され、コンテナは、セキュリティサービス呼び出しにおける後のほうのステップで、それらを使用する。いくつかの実施形態では、トークンは、クライアント自身の認証された識別情報を表す。クライアントは、セキュリティサービス712(セキュリティサービス、又はセキュリティサービス314と同じ)に、「このトークンによって識別されるユーザーは、アクセス、及びターゲットリソース上で要求された能力を実行できるか？」という質問714を投げ掛ける。

【００６１】

この質問に答えるために、セキュリティサービスは、図７においてAC716として示されるアクセス制御装置を使用する。AC716は次の２つのことを行う：それは、一連のアクセス判定(AD)プラグイン718,720,722全体に渡って、分散させる端末増設機構として動作し、それはまた判定器としても動作する。ACがアクセス判定を呼び出す時、それは「これはOKですか？」と尋ねながら、最初のアクセス判定718から順番に、トークンを各アクセス判定に回す。アクセス判定は、許可、拒否、又は保留という三つの返答のうちの一つを返す。もし次のアクセス判定があれば、アクセス制御装置は次のものに移る、というように全てのアクセス判定がポーリングされるまで行う。実際には、セキュリティ機構は、一連の窓ガラスのように動作する。もしX718が「許可」といえば、処理は続行する；もしX718が「拒否」といえば、処理は止まる。ACは、次にアクセス判定Y720へ移り、そして全く同じ識別情報を提示しながら、全く同じ質問を尋ねる、そしてY720も同様に、許可、拒否、又は保留で返答する。

【００６２】

この処理は、サービスがこれらプラグインをどれだけ多く有していても、それらに対して続けられる。判断ポリシーは、もし何れか一つが拒否と言えば、処理を終了させる、又は、単に全体拒否を返す、ということを定めている。アーキテクチャは、もしX718が許可、及びY720が拒否、及びC722が保留であるならば許可とみなすような、充分に柔軟な判定を行う判断ポリシーを考慮している。他の実装は、もっと強硬路線を採る：もしどれか拒否するものがあれば、それは全体拒否とする。ACが、アクセス判定に対しアクセス判定が一義的に理解できない質問を尋ね、それゆえ明確な判定をできない時には必ず、保留が返される。これは、従来の承認システムが含まれるときは常に、最も一般的なものであり、全てではないが幾つかの動作、又はX718が企業ポリシーでありかつYがビジネス路線である時に、必要とされる。この場合、ビジネス路線ではなく、企業ポリシーによって実行されるような質問が尋ねられると、ビジネス路線の判定器は「これを理解できない、保留する」と言う。

【００６３】

全てのアクセス判定器がポーリングされ、要求が最後に通過する点までどれも拒否を示さ

なければ、その時にはアクセス要求はどれも拒否しない、あるものは許可しかつあるものは保留する、例えばY720は保留しかつX718及びC722は両方とも許可、という点に到達する。次に判断ポリシーは、どのように結果を定めるかを制御するために使用することができる。ある一実施形態では、コンテナに対し、許可又は拒否という二つの結果のみがある。そこでアクセス制御装置は、どのようにして判定するかを決める判断ポリシー見る。たとえば、ある場合には、そのボックスの実装より、システムは、「全員一致の許可を要求しますか？」と言う判断ポリシーを使用するが、この場合、X、Y、及びCすべてが許可しなければならない。もしどれか一つでも許可しなければ、結果は拒否となる。その他のアプローチは、全員一致の同意を必要とするものではなく、その場合、どれかが拒否と言わない限り、アクセスは許可される。この判定は、判断ポリシーに基づいてなされる。

【００６４】

アクセス判定の過程の中で何が起きたとしても、セキュリティサービス自身は、何が起きたのか及びその結果の検査を求めて、要求を検査サブシステム724に渡す。その点においてのみ、セキュリティサービスは、コンテナに最終結果が許可又は拒否であるかを教える。もし許可であるなら、コンテナは、要求された操作が保護されたリソースへ送られることを可能にする。もし拒否であるなら、要求はその点まで届かず、即座に拒絶される。

【００６５】

本発明はまた、例えば後述の保護されたリソース上でのアクセス要求の正常な受け取り等、の途中で、アクセス判定過程を呼び出す方法を提供する。このシナリオでは、システムは要求に対する最初のチェックを通過し、かつ全てが許可され、そこでその方法がリソースへ送られたことを前提としている。リソースは、クライアントに戻すデータを返す。ただやみくもに、このデータを返すのではなく、むしろ、承認チェックが再実行され、今度はセキュリティサービス、特にアクセス判定が、そのデータがクライアントに返されるのがOKであるか否かを判定するために、返されるデータを調べる。ここでの一例は、ユーザーがドキュメントを要求しようとするビジネス操作の例である。ユーザーは最初に入るとき、彼らに割り当てられたセキュリティ許可を持ち、それは、ユーザーが、彼らのセキュリティ許可上で機密扱いとされないドキュメントを見ることができることを意味する。もし、彼らが秘密セキュリティ許可のみを持ち、かつ返されるはずのドキュメントが最高機密である場合には、たとえ彼らがそれを実行するための手段を持ち合わせていたとしても、システムは、出力をユーザーに戻すことを許さないが、その理由は、もしそれを許せば、ユーザーは見ることを許されないものを、見ることが可能になってしまうからである。アクセス判定は、今度はユーザーに戻す結果を見直しながら、再実行され、及び同種の判定を行う。この例では、例えばYアクセス判定720が「出力が何であろうと関係ない。入ってくる入力にのみ関心がある」と言うような、多くの保留が存在する。開発者は、ビジネス路線に基づいて、又は企業ポリシーに基づいて、異なるポリシーを実現させるために、プラグインを組み合わせ、及び整合させることができる。

【００６６】

図８は、安全性の確保された、又は保護されたリソースへのアクセスの提供において、本発明の実施形態で使用される過程を示すフローチャートである。ステップ802で、クライアントはコンテナ内部で、アプリケーションを開発し、及び配置する。アプリケーションが保護されたリソースへのアクセスを要求するとき、呼び出しが作成され、ステップ804で示されるように適切なコンテナへ送られる。コンテナは、ステップ806で、セキュリティサービスを呼び出すために使用される。セキュリティサービス内では、ステップ808で、端末増設機構のアクセス制御装置が、この特定のアクセス要求を検証、又は確認するにあたって、どのアクセス判定が起こるべきかを判定する。ステップ810では、各アクセス判定が選択され、及びアクセス要求へのその寄与判定のために処理又はポーリングされ、各アクセス判定の結果は拒否、許可、又は保留となる。検査機構は、ステップ812で、各アクセス判定の結果の跡をたどる。アクセス判定の結果及び累積的な応答--それは許可、拒否、又は保留である--に基づいて、ステップ814では、コンテナは、保護されたリソースへのアクセスを与えられる(又は拒否される)。ステップ816では、データは次に、クラ

イアントアプリケーションから保護されたリソースに、及び、逆に保護されたリソースからクライアントアプリケーションに、渡される。もし、クライアントがその後、同じリソース又は異なるリソース（ステップ818で判定されるように）に対して要求する場合には、その要求は適切なコンテナへ送られ、及び他の承認判定が要求され(ステップ820)、そうでなければ、セッションは終了する(ステップ822)。

【００６７】

図９は、権限付与レイヤー904、及びビジネスアプリケーションレイヤー908に対するその位置付け、ビジネスポリシーレイヤー906及びセキュリティ承認レイヤー902を含む、本発明の一例に従ったセキュリティ及び個人化レイヤーを示している。当業者には、図９に示される説明図が、どのように権限付与レイヤーが開発されるかの概念表現であること、そしてこのアーキテクチャについて多くの他の変形を使用できるということよりもむしろ、このような個々のレイヤーをどれか特定のデザイン実装の中で使用することはできない(実際に殆ど使用されない)ということが、明らかであろう。図９は、権限付与904がどのように、従来のセキュリティアーキテクチャ(902として表現される)と、しばしば別個であるビジネスポリシーアーキテクチャ(906として表現される)との間の境界を越えるのに使用されるか、を示している。このような権限付与の使用を通して添えられる情報は、例えば会社のビジネスアプリケーションを個々の実体のために、その実体の権限付与に基づいて個人化するというような、908として表現される会社のビジネスアプリケーションの実施、を導くために使用される。

【００６８】

図10は、権限付与が、動的に役割を生成するために、どのように使用されるかを示している。権限付与処理1002は、セキュリティの質問がある状況の中で尋ねられることを可能にする。本発明は、その権限付与に基づいて、特定の対象1000、例えば主体１1004、主体２1006から主体ｎ1008までに、動的に役割を割り当てる能力を備え、これはさらに、システムがセキュリティレイヤーの中でアプリケーションの状況を提供することを可能にし、及び承認質問がアプリケーションの観点から表現されることを可能にする。医者と患者の一般的な例のように、質問が状況の観点から表現されるならば、システムはそれらの質問に正確に答えるだけである。もし質問が、「スミス医師は、患者のカルテを修正できますか？」と投げ掛けられたら、それは状況なしの質問である。従来の答えは、「ある時はイエス、及びある時はノー」である。厳密に答えを判定する唯一の方法は、例えばどの患者について話しているのか、あるいはどのカルテについて話しているのかというような、質問が尋ねられた状況を知ることである。システムはその時、所有者のように役割を割り当てる、又は、システムがセキュリティ判定ではなく、むしろビジネス判定を可能とするような、動的な割り当てを基本的に行う。

【００６９】

単純な表記を用いて、この概念を次のように表現することができる：

各対象において、アクティブな役割ARは、その対象が現在使用しているものである：

AR(s:対象)＝{対象sにおけるアクティブな役割}

各対象は、一以上の承認された役割RAと関連付けられる：

RA(s:対象)＝{対象sについて承認された役割}

各役割は、一以上のタスクTAを実行するために、承認される：

TA(r:役割)＝{役割rについて承認されたタスク}

もし述語関数exec(s,t)が、現在の値で、時間内に、現時点で真であるならば、対象はタスクを実行する：

$$exec(s:\text{対象},t:\text{タスク})=\text{真} \Leftrightarrow \text{対象}s\text{はタスク}t\text{を実行することができる}$$

【００７０】

上記の記述が正しく評価を行うために、一組のルールが要求される：

1.役割割り当て　-　役割を割り当てられた場合のみ、対象はトランザクションを実行することができる。システム上で実行される全ての他の動作はタスクを通して行われるが、

識別及び認証の過程はタスクの一部と見なされない：

$$\forall s:対象, t:タスク(exec(s,t)AR(s) \Rightarrow \neq 0)$$

2.役割承認 - ある対象のアクティブな役割は、その対象について承認されなければならない。このルールは、ユーザーが承認された役割のみを引き受けることができることを、保証するものである：

$$\forall s:対象(AR(s) \subseteq RA(s))$$

3.タスク承認 - 対象は、タスクがその対象のアクティブな役割について承認された場合のみ、そのタスクを実行することができる：

$$\forall s:対象, t:タスク(exec(s,t) \Rightarrow t \in TA(AR(s)))$$

【００７１】

別の例として、開発者が、ユーザーに権限付与されている事に基づいてウェブページを個人化するために、本発明を使用することができる。たとえば、自己登録又は自己支援機能は、クレジットカードのウェブページの一部とされることが可能であり、そこではユーザーが、クレジットカードナンバーを変更することができる唯一の者である。ベンダーは、それらを読むことは出来るが、変更することはできない、また他の誰もそれらを見ることは出来ない。システムは、その状況を知らなければ、これをどのようにして行うか解らない。誰かクレジットカードフィールドを修正することができるか？もちろん、誰かはクレジットカードフィールドを修正することができる。ではそれならば、誰がそれを修正することができるのか？如何なるメンバーも、他のメンバーのクレジットカードを修正することは許されないので、その時システムは、どの特定の一人であるか尋ねなければならない。そのためシステムは、どのオブジェクト及びどのプロファイルが審議中であるか、を知らなければならず、そこで「あなたは、そのプロファイルの所有者ですか？」と尋ねる。従来は、定義される唯一の事が、静的な役割であった。本発明は、権限付与の利用を通じて、ビジネスアナリスト及びビジネスポリシー作成者が、オブジェクト又は主体の識別情報を定義するために、実行時に計算される動的役割を定義することを可能にする。これらの役割は、サーバーアーキテクチャの基礎となっているセキュリティ仕様に違反しない。例えば、要求は、所有者の役割として登録された呼出元となることもできる。システムは所有者の役割を定義しないが、しかし所有者の役割は動的にユーザーと関連付けられることが可能であり、及びユーザーがもはや所有者ではないという何らかの行為を行うや否や、ユーザーから無効とさせることもできる。EJBのような技法は、何らかの行為を行うために、その使用が許されるべきかどうかという事が、あなたがその役割にあるか否かということに基づき、かつあなたがこの可能性を問い合わせるために標準EJBを使用する、という点でこれを利用することができる。

【００７２】

本発明がどのように使用されるかの他の例は、一回限定ログオン機構の設備を通じたものであり、これは従来のセキュリティ実装の要素と、ビジネスポリシー実装の要素を組み合わせるものである。図11に示されるように、第一のセキュリティ領域Ａ1102、例えば第三者アプリケーションによって提供され、及びユーザープロファイル又は企業のビジネスポリシーの情報を全く持たないセキュリティ領域での、最初のログイン処理の間に受け取られた情報は、第一のログインモジュール1108によって取り込まれ、及び第二のセキュリティ領域1104の中にある二番目のログインモジュール1110に渡される（矢印1104で示されるように）。第二のセキュリティ領域Ｂ1104は、ビジネスポリシーの情報を持ち、かつ元のログイン情報に加えて、ユーザーに対する権限付与又は権限付与プロファイル1102を判定又は計算することができる。元のログイン情報及び権限付与情報は、第三のセキュリティ領域Ｃ1106の中にある三番目のログインモジュール1112に渡される(それぞれ、矢印1108及び矢印1120で示されるように)、というように続く。

【００７３】

セキュリティプロバイダーインターフェース
本発明と共に使用されるセキュリティプロバイダーインターフェース(SPI)は、次に挙げるものについてのインターフェースを含む：
・認証 - Java認証及び承認サービス(JASS)ログインモジュール
・承認 - アクセス判定
・検査 - 検査チャネル
・主体->役割マッピング - JAASログインモジュール
・主体->証明マッピング - JAASログインモジュール
・鍵記憶装置 - Java2鍵保管Spi
・証明書検索及び無効化 - Certパス
・証明書マッピング - Cert認証装置
・ハードウェア加速器 - Java暗号化拡張
・周辺保護 - 接続フィルター

【００７４】
ここで記述されるSPIは、本発明と共に使用されるインターフェースの形式の単なる例にすぎず、かつ本発明によって提供されるセキュリティサービスアーキテクチャの柔軟性を説明する目的でここに列挙されたものであることは、当業者には明白であろう。本発明の技術的範囲内である限り、他の形式のインターフェースが、ここで記述されるインターフェースと置き換えられる、又は強化するということも可能である。

【００７５】
認証SPI
本発明で使用されるJAASログインモジュールは、標準JAASログインモジュールに基づくものであるが、しかしさらに、標準ログインモジュールの実装をサポートする。認証SPIは、ユーザー、グループの管理をサポートし、及び複数のJAASログインモジュールのサポートを提供するまでに、拡張される。資格は、ログインモジュールのインスタンス全体で共有され、かつユーザープロファイル情報を用いて更新する個々のログインモジュールが提供される。認証SPIの任務は、セキュリティ領域範囲及び、JAAS対象内の主体群に基づくユーザー認証を含む。

【００７６】
承認SPI
承認SPIはアクセス判定に備えるものであるが、これに対応するJavaの全くの等価物は存在しない。承認SPIは、宣言及びルールの両方をベースとした、様々な承認モデルをサポートする。呼び戻し処理装置は、呼び出し状況へのアクセスを獲得するために使用される。SPIはまた、主体の識別情報、グループの帰属資格、及び役割割り当てといった対象情報、及びユーザープロファイル情報(任意)へのアクセスを提供する。多重アクセス判定プロバイダーは、スタックの中で構築される。承認SPIの任務は、許可、拒否、又は保留の承認判定を行うこと、又はアプリケーションの適用範囲内である保護されたリソースへのアクセス要求を作成することを含む。図11は、ドメイン内のセキュリティ領域間に、プロファイル情報を分散させるために、本発明がどのように使用されるか、一例を示している。

【００７７】
検査SPI
検査SPIは検査チャネルを含むものであるが、これに対するJavaの等価物は何も定義されていない。呼び戻し処理装置は、対象情報、主体の識別情報、グループの帰属資格、及び役割割り当てへのアクセス、及びユーザープロファイル情報(任意)を含む呼び出し状況へのアクセスを獲得するために使用される。再度、多重検査チャネルプロバイダーがスタックの中で構築される。検査SPIの任務は、情報が検査されるべきか否かを判定することと、及びQoSポリシーに基づいてデータの実際の検査を行うことを含む。

【００７８】
主体－＞役割マッピングSPI

主体－＞役割マッピング(JAASログインモジュール)SPIは、JAASログインモジュールに基づくものであり、かつ、役割を動的に、対象内に含まれる主体の識別情報にマッピングするために使用される。役割は、管理者によって、又は配置記述子から明確に定義されたもの、又はビジネス要求のパラメータ、主体のプロファイル中の属性値、並びにその他の条件に基づいて動的に計算されるものである。

【００７９】

主体－＞証明マッピングSPI

主体－＞証明マッピング(JAASログインモジュール)SPIは、JAASログインモジュールに基づくものであり、かつセキュリティドメインポリシー又はセキュリティドメイン技術の境界を越える時に、主体の識別情報をマッピングするために使用される。主体マッピングSPIの任務は、提供された対象に基づくものであり、かつユーザー名/パスワードにおけるパスワード証明、及びトークン形式の証明における汎用的証明のような、適切な情報を有する公的証明を対象に付加するために使用される。

【００８０】

鍵記憶装置SPI

Java2鍵保管SPIは、Java2鍵保管インターフェースに基づくものであり、かつファイルベースの媒体:(PKCS #5/#8, PKCS #12)、HSMベースの媒体:(PKCS #11)、スマートカード/Javaカードといった様々な媒体内に保管される鍵への一貫したアクセスを提供する。鍵保管SPIは、暗号署名に必須なサポートを提供し、かつその任務は、保護されたリソースからの秘密鍵検索、及び保護された記憶装置からの機密検索を含む。

【００８１】

証明書検索SPI

証明書検索SPI(Certパス)はJSR55に基づくものであり、かつ複数の証明書形式のサポートのほか、デジタル証明書の検索、検証、及び無効化のための一貫した機構を提供する。

CertパスSPIの機能は、次のことを含む：

Cert保管:問い合わせを介して、記憶装置から証明書の検索；

Certパス：Cert保管を利用した、証明書の連鎖の検索；

Certパスチェッカー：証明書を確認する；

Certパス確認装置：証明書の連鎖を確認する

【００８２】

証明書マッピングSPI

証明書マッピングSPI(Cert認証装置)についてJavaの等価物は定義されていないが、この証明書マッピングSPIは、セキュリティドメイン内での、デジタル証明書と主体識別情報との間のマッピングを可能にするための拡張可能な方法を提供する。それはさらに、Cert認証装置の呼び出しの前に実行される、検証及び無効化のチェックを提供する。

証明書マッピングSPIの任務は、検証されたクライアントのデジタル証明書に基づくものであり、それをユーザー名及び主体領域にマッピングする。

【００８３】

ハードウェア加速器SPI

ハードウェア加速器SPIはJava暗号化拡張を使用し、かつJava2セキュリティアーキテクチャに基づくものである。その任務は、セキュリティサービスの制御下で暗号化サービスを提供することを含む。

【００８４】

周辺保護SPI

周辺保護SPIは、組込みの接続フィルターの実装を提供するための接続フィルターを含む。これは使い易さに的が絞られ、かつMbeansを介してコンソールから設定できるものであり、それぞれ順番に実行されるユーザー作成フィルター、及び縦続接続フィルターの継続したサポートを可能とする。これは、顧客が運用しなければならない条件を最小化するものである。

【００８５】

セキュリティ管理SPI
セキュリティ管理SPIは、第三者ベンダーのコンソールを、サーバーコンソールに統合させ、及びセキュリティが、これらベンダーからサーブレットへ転送することを可能にする。セキュリティ管理SPIは、ユーザー、グループ、役割、及び承認情報の完全なライフサイクルのサポートを提供する。
【００８６】
一回限定ログオン(SSO)SPI
一回限定ログオンSPIは、手続開始主体の識別情報からリソース識別情報へのマッピングを行い、及びリソース証明の安全な記憶領域を提供するために、ログインモジュールを使用する。
【００８７】
本発明のより好ましい実施形態についての上述の説明は、例証及び説明の目的のために提供されたものである。それは網羅的なもの、又は本発明を開示された厳密な形態に限定しようとするものではない。多くの修正及び変更が、当業者には明らかになることが明白であろう。本実施形態は、本発明の原理、及びその実用例を最もよく説明するために選択され、かつ記述されたものであり、それによって当業者は、意図された個別の利用に適した様々な実施形態における、及び様々な修正を伴った本発明を理解することができる。本発明の技術的範囲は、次に述べる特許請求の範囲及びその均等技術によって、定められるものである。
【図面の簡単な説明】
【００８８】
【図１】従来の技術知識に従った、クライアント/サーバーのアーキテクチャの説明図を示している。
【図２】本発明に従った、クライアント/サーバーのアーキテクチャの説明図を示している。
【図３】本発明に従った、セキュリティサービスの説明図を示している。
【図４】本発明に従った、セキュリティサービスで使用される方法のフローチャートを示している。
【図５】本発明に従った、サーバーセキュリティの配置ツール及び処理の説明図を示している。
【図６】本発明に従った、サーバーセキュリティの配置ツールで使用される方法のフローチャートを示している。
【図７】本発明に従った、セキュリティサービス及び保護されたリソースの説明図を示している。
【図８】本発明に従った、保護されたリソースへのアクセスを許可するためにセキュリティサービスで使用される方法のフローチャートを示している。
【図９】本発明の実施形態に従った、セキュリティレイヤーの説明図を示している。
【図１０】本発明の実施形態に従った、権限付与の機構の説明図を示している。
【図１１】本発明に従った、いくつかのセキュリティドメインわたるログインの説明図を示している。

【図１】

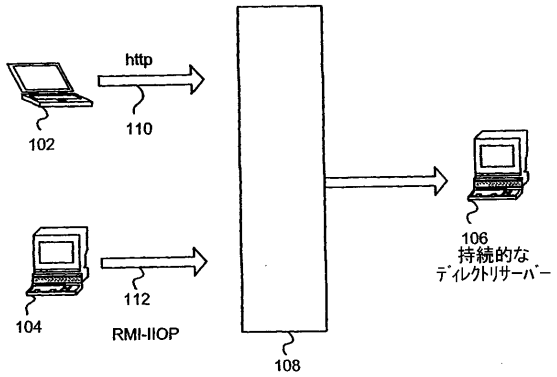


FIGURE 1

【図２】

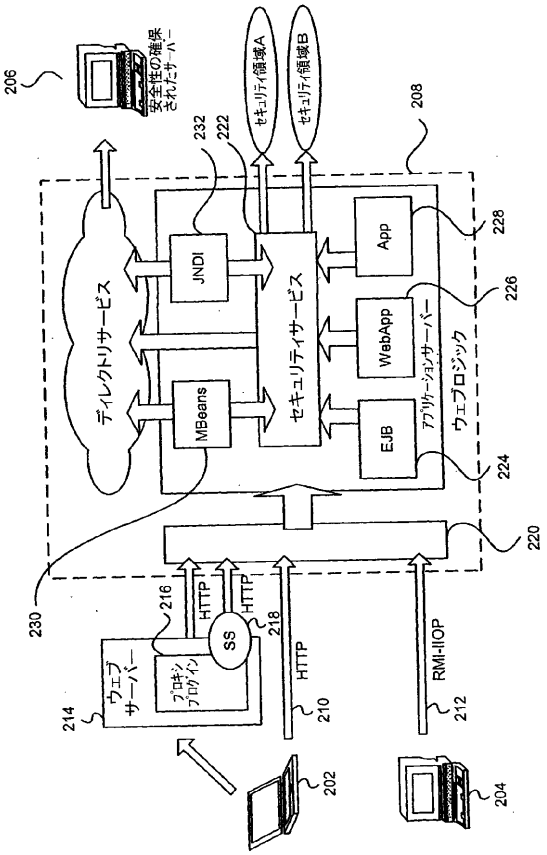


FIGURE 2

【図３】

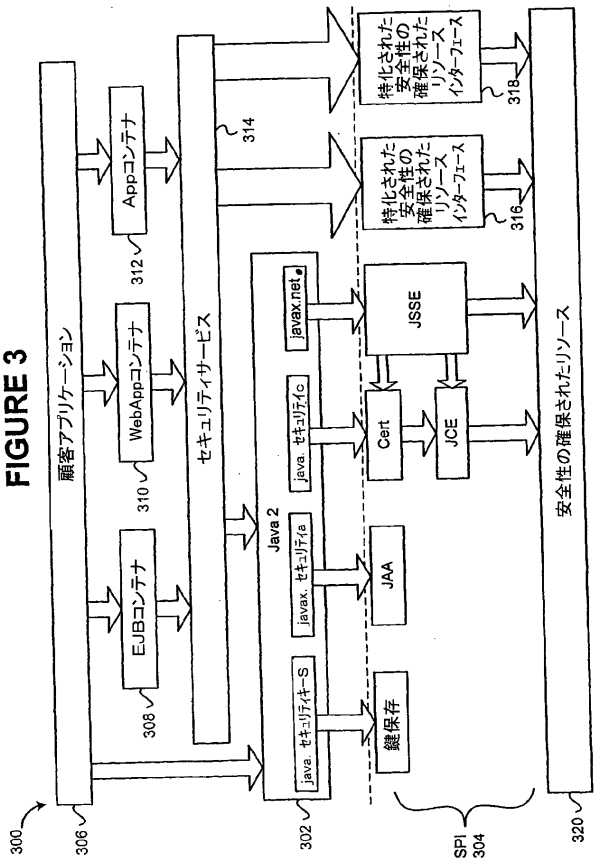


FIGURE 3

【図４】

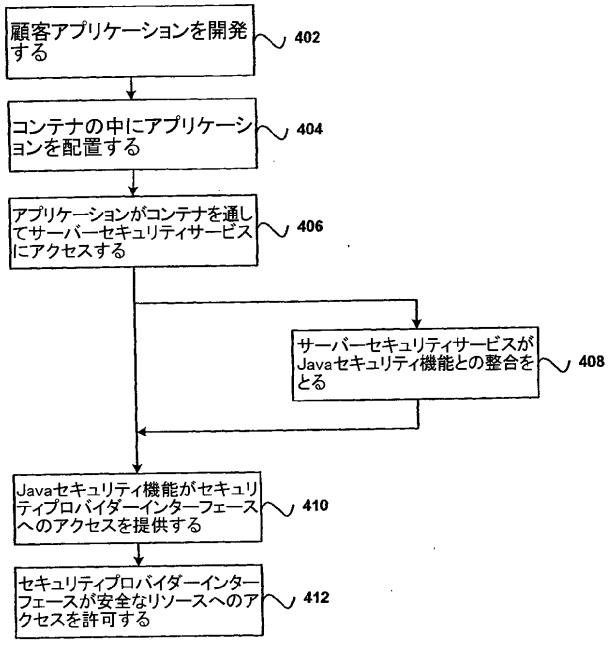


FIGURE 4

【図５】

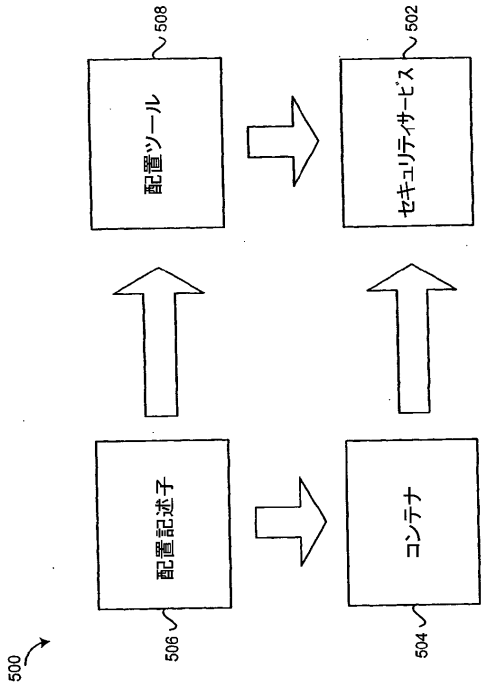


FIGURE 5

【図６】

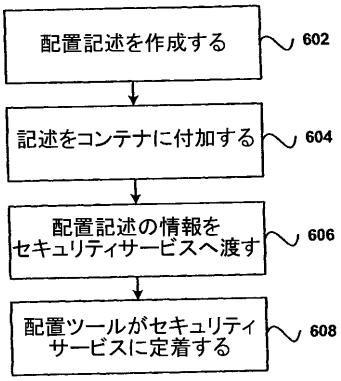


FIGURE 6

【図７】

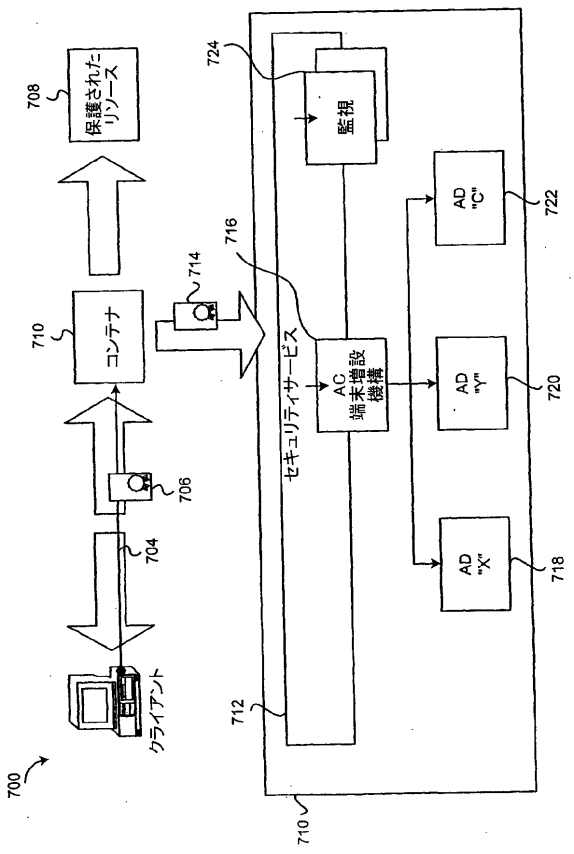


FIGURE 7

【図８】

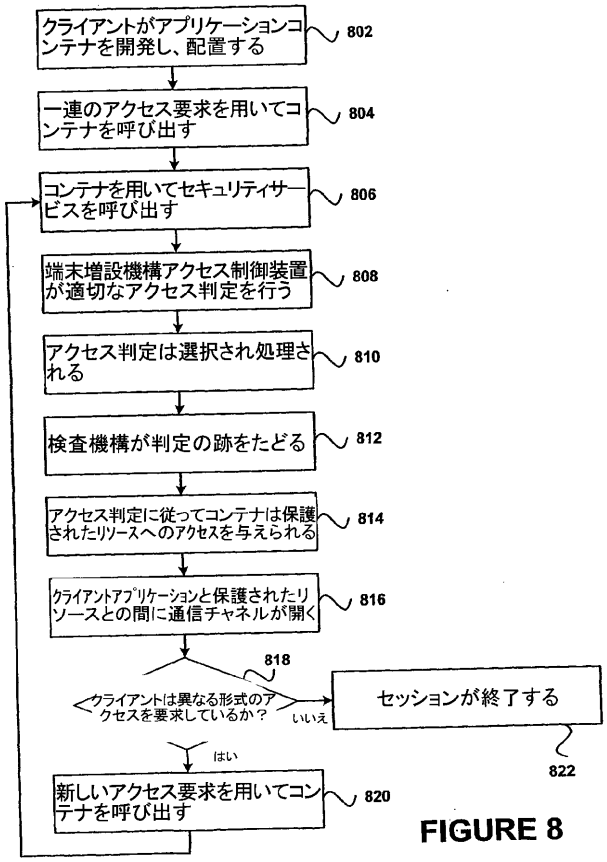


FIGURE 8

【図９】

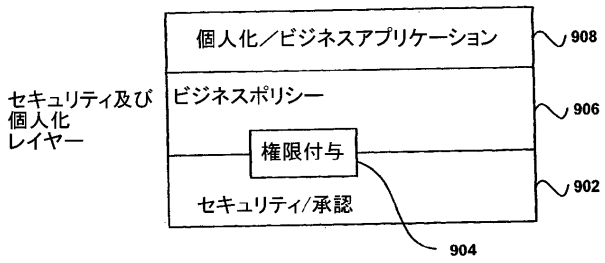


FIGURE 9

【図１０】

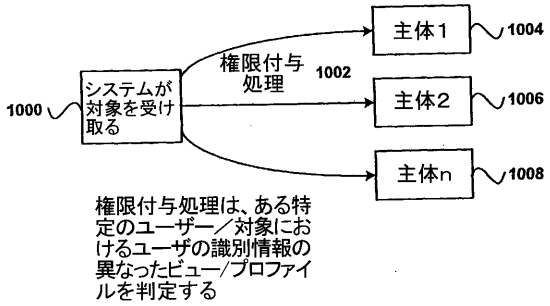


FIGURE 10

【図１１】

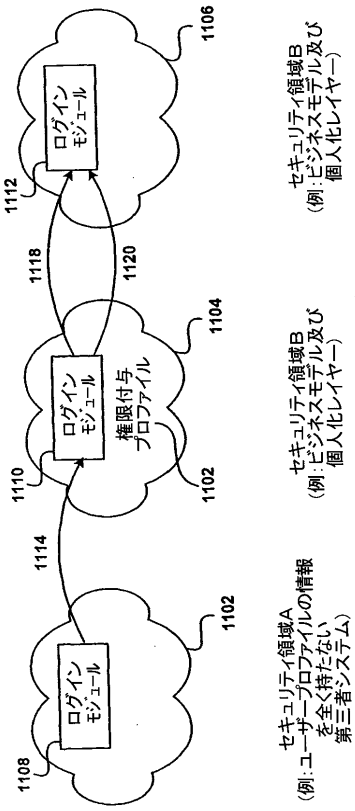


FIGURE 11

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 December 2002 (19.12.2002)

PCT

(10) International Publication Number
WO 02/101973 A1

(51) International Patent Classification⁷: H04L 9/00

(21) International Application Number: PCT/US02/16644

(22) International Filing Date: 29 May 2002 (29.05.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/878,536 11 June 2001 (11.06.2001) US

(71) Applicant (for all designated States except US): BEA SYSTEMS, INC. [US/US]; 2315 North First Street, San Jose, CA 95131 (US).

(72) Inventor: PATRICK, Paul; 9 Cobblestone Lane, Manchester, NH 03109 (US).

(74) Agents: MEYER, Sheldon, R. et al.; Fliesler Dubb Meyer and Lovejoy LLP, Four Embarcadero Center - Fourth Floor, San Francisco, CA 94111-4156 (US).

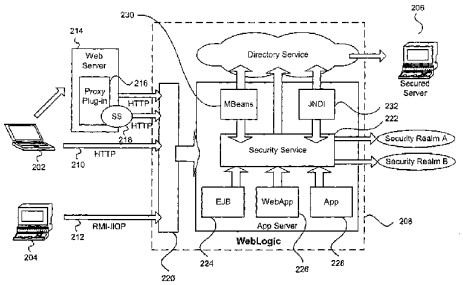
(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
with international search report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD FOR SERVER SECURITY AND ENTITLEMENT PROCESSING



(57) Abstract: A pluggable architecture allows security and business logic plugins to be inserted into a security service hosted by a server (206), and to control access to one or more secured resources on that server, on another server within the security domain (208), or between security domains. The security service (222) may act as a focal point for security enforcement, and access rights determination, and information used or determined within one login process can flow transparently and automatically to other login processes. Entitlements denote what a particular user (204) may or may not do with a particular resource (226), in a particular context. Entitlements reflect not only the technical aspects of the secure environment (the permit or deny concept), but can be used to represent the business logic or functionality required by the server provider. In this way entitlements bridge the gap between a simple security platform, and a complex business policy platform.

WO 02/101973 A1

SYSTEM AND METHOD FOR SERVER SECURITY AND ENTITLEMENT PROCESSING

Claim of Priority:

[0001] This application claims priority from United States patent application entitled "SYSTEM AND METHOD FOR SERVER SECURITY AND ENTITLEMENT PROCESSING" Application No. 09/878,536 filed June 11, 2001 and which application is incorporated herein by reference.

Field of the Invention:

[0002] The invention is related generally to server security mechanisms, and specifically to an architecture for server security.

Background of the Invention:

[0003] Over the years the term "security" as applied to server technology, and particularly to e-commerce servers, has expanded to cover several aspects of a secure environment. The original definition of the term has always envisaged a mechanism for validating a particular user's (or client's) identity, and preventing that user from retrieving, viewing, or otherwise accessing, information on the server which they are not allowed to access.

[0004] **Figure 1** shows a typical example of a security mechanism commonly used today. Clients **102, 104** access a secured resource such as an application server **106** via a number of protocols; including for example http (hypertext transfer protocol) **110** and IIOP (Internet Inter-

ORB protocol) **112**. The access requests are filtered through a security layer **108** which determines, typically based on a set of hardwired rules, whether to allow or disallow the requested access. The security layer may be a component of the protected resource (e.g. the application server itself) or may operate as a separate entity (for example as part of a firewall system or device).

[0005] With current systems little or no attempt is made to analyze the nature of the access request, the type of client, or the type of protected resource. As such, security is often thought of as a simple "permit or deny" mechanism, designed in such a way as to understand the protected resources provided by the server, and to adhere to a predefined set of rules governing a user's or a client's access rights to those resources. There is little or no understanding of the manner in which the request is made, or the environmental setting in which a particular user may make a request to access a resource, and the security mechanism does not lend itself to easy modification of the rules to reflect new changes in business policy regarding security.

[0006] Attempts have been made to develop the concept of security and particularly server security to include information that reflects a user's particular environment, and the manner in which a request to access a particular secure resource is phrased. Similarly, attempts have been made to provide security mechanisms that can be easily modified to reflect changes in business policy, security policy, and access rights. These methods still typically require an application programmer to be responsible for assembling a set of business policy queries together and asking the question, "is this OK ?-- is this access acceptable according to the security rules?" What is needed is a mechanism to make this process all inclusive so that the programmer does not have to understand the business semantics and the business policy rules in order to be able to develop programs and applications that ask security related business policy questions.

[0007] Recently, a new set of requirements concerning the securing of resources has emerged from discussions with application server customers and system integrators. These new requirements are expressed in terms of security from the point of view of an application instead of infrastructure. A key differentiation within these new requirements is the existence of business policy enforcement. The enforcement of business policy is typically described in terms of actions that a user is "entitled" to perform based on the role in which they are acting and the context of the business request. Customers and system integrators are frustrated with the fact that they are required to embed code that enforces business policy within applications. Embedding this type of logic creates deployment problems, in that the application must be modified, tested, and re-deployed each time the business policies are changed. Given the rate at which business policy changes, the current requirements for modification and re-deployment are unacceptable.

[0008] Customers and system integrators would ideally like to have these new authorization capabilities universally applied across the different types of execution and resource containers, including for example those for Enterprise Java Bean (EJB), Web Applications (Servlet, JSP), as well as other types of business and resource containers. Role-based access control (RBAC) is becoming one of the primary form of authorization requested by customers and system integrators. The trend in use of roles allows organizational identity to be used as an abstract form of identity when making authorization decisions. The use of roles provides a mechanism to reduce both the cost and errors in the administration of application security since it simplifies the amount of administration.

[0009] The Java 2 Enterprise Edition (J2EE) defines a declarative form of role-based access control. However, because it is based on a declarative scheme, the association of roles to principals is static. In addition, the current Java 2 Enterprise Edition specification provides no

means by which the context of a business request, such as the parameters of the request or the identity of the target, can be taken into account when determining the roles to be associated with a given principal. Consequently, application developers are required to implement business policy rules within the application to compute dynamic roles associations in order to support concepts like "owner".

[0010] Customers and system integrators nowadays demand a richer set of authorization capabilities than those provided with the permission-based security defined by Java 2 security and the role-based access control defined in Java 2 Enterprise Edition. The foundation of the requirements for this richer level of authorization can be found in the intersection of classical authorization mechanisms, such as Access Control Lists, and business policy enforcement; and include the ability to take the context of the business request into consideration when making an authorization decision. The request context may include the identity of the target object, the value of the parameter of the request, and potentially environmental information such as the network or IP address of the initiating client.

[0011] The lack of a single mechanism through which to integration these new authorization capabilities, regardless of execution or resource container type, is a point of frustration with customers and system integrators. The Service Provider Interface (SPI) is the mechanism used by several application servers, including the WebLogic Server product from BEA Systems, Inc., San Jose, California, to allow integration with external authorization providers. This SPI realm has a number of limitations that limit it's ability to be used as a successful means to integrate 3rd-party authorization mechanisms, or new authorization capabilities being required by customers and system integrators.

[0012] One of the largest limitations with the current "realm" SPI is scope of enforcement. Currently, the enforcement scope of the realm mechanism does not cover resources such as Enterprise Java Bean and

Web Applications. In particular, the realm is focused on RMI style resources such as JMS destinations, entries in JNDI, and servlets at a course-grained level. While the realm could conceivably be updated to hold the definition of the authorization policies required to support protection of such resources, it is the other limitations that ultimately make the realm an unrealistic mechanism to address all the authorization requirements.

[0013] The second limitation of the current realm SPI is point of enforcement. The current realm mechanism does not allow the point at which the decision is made to allow access to a protected resource to exist within the realm itself. Instead, the point of enforcement is within the application server itself. Because of this approach, the current realm is defined to support only a permission-based authorization mechanism where the realm simply acts as a database of Access Control Lists. The complexities of providing a Java 2 Policy object that can also support the Java 2 sandbox rules is unacceptable to most vendors.

[0014] Yet another limitation is the Enforcement Mechanism Support of the current realm SPI. As with the point of enforcement, the enforcement mechanisms allowed by the current realm SPI are limited to those based on permission-based authorization mechanisms. This is counter to the capabilities of the leading 3rd-party authorization providers, the integration of which is being requested by customers and systems integrators on a daily basis.

[0015] Together, these limitations constrain the types of authorization providers that can be integrated with an enterprise application server without minimizing the value proposition of the provider. There remains no current capability to obtain the context of the request in order to provide the rich authorization requested.

Summary of the Invention:

[0016] The invention is related generally to server security mechanisms, and specifically to an architecture that provides for server security and entitlement processing. A pluggable architecture allows security and business logic plugins to be inserted into a security service hosted by a server, and to control access to one or more secured resources on that server, on another server within the security domain, or between security domains. The security service may act as a focal point for security enforcement, and access rights determination, and information used or determined within one login process can flow transparently and automatically to other login processes.

[0017] The invention also introduces the concept of entitlements that are used within an access context. As used in the context of this application, a "user" or a "client" may refer to the same thing - either a physical person, a hardware device or software application under control of the physical person, or a hardware device or software application operating under autonomous control without user intervention. The user (or client) is the entity which is trying to access a protected resource on the server. This protected resource may, for example, be a software application running on the server, a particular Web page or portion of a Web site, or a database etc.

[0018] When the user attempts to access the resource, the security service may determine the type of access request, the destination (protected resource), and the setting in which the request is made - hereinafter referred to as the access context or simply the context. From this information the security service can determine an "entitlement", or a set of entitlements, for the user. Entitlements clearly denote what a particular user may or may not do with a particular resource, in a particular context. Entitlements reflect not only the technical aspects of the secure environment (the permit or deny concept), but can be used to represent the business logic or functionality required by the server

provider. In this way entitlements bridge the gap between a simple security platform, and a complex business policy platform.

[0019] To illustrate the capability, consider the following business example:

The answer to the question "Can Dr. Smith update a patient's medical chart" is dependent upon the context in which the question is asked. In a permission-based authorization system, this context is absent since the resource is some instance of a 'medical chart' object, the request is to 'update', and the Subject is 'Dr. Smith'. Consequently, if the answer rendered is 'Yes', then Dr. Smith could update any patient's medical chart. In a capabilities-based authorization system, it is possible to add the necessary context of who's the patient in question. Thus, the question can now be rephrased as "Can Dr. Smith update Jon Joe's medical chart?" In determining the answer to this question, we now need to know if Dr. Smith is Jon Joe's personal physician, or perhaps an attending physician at a medical center. Using a simple rotation we can represent the concept as follows:

[0020] In one embodiment, the invention comprises a security system for allowing a client to access a protected resource, comprising an application interface mechanism for receiving an access request from a client application to access a protected resource, and communicating said access request to a security service; a security service for making a decision to permit or deny said access request; and a resource interface for communicating permitted access requests to said protected resource.

[0021] In another embodiment the invention comprises a method of allowing a client to access a protected resource, comprising receiving at an application interface mechanism an access request from a client application to access a protected resource and communicating said access request to a security service; making a decision at said security service to permit or deny said access request; and communicating via a resource interface a permitted access request to said protected resource.

Brief Description of the Figures:

[0022] **Figure 1** shows an illustration of a client/server architecture in accordance with the prior art.

[0023] **Figure 2** shows an illustration of a client/server architecture in accordance with the invention.

[0024] **Figure 3** shows an illustration of a security service in accordance with the invention.

[0025] **Figure 4** shows a flowchart of a method used by the security service in accordance with the invention.

[0026] **Figure 5** shows an illustration of a server security deployment tool and process in accordance with the invention.

[0027] **Figure 6** shows a flowchart of a method used by the server security deployment tool in accordance with the invention.

[0028] **Figure 7** shows an illustration of a security service and a protected resource in accordance with the invention.

[0029] **Figure 8** shows a flowchart of a method used by the security service to allow access to a protected resource in accordance with the invention.

[0030] **Figure 9** shows an illustration of a security layer in accordance with an embodiment of the invention.

[0031] **Figure 10** shows an illustration of an entitlement mechanism in accordance with an embodiment of the invention.

[0032] **Figure 11** shows an illustration of a login across several security domain models in accordance with the invention.

Detailed Description:

[0033] An embodiment of the invention includes a security architecture that provides for server security and entitlement processing, that allows security and business logic plugins to be inserted into a security service hosted by a server, and that can be used to control access to one or more secured resources on that server, on another

server within the security domain or realm, or between security realms. The security service acts as a focal point for security enforcement and access rights determination, and information used within one login process can flow automatically to other login processes, allowing for single sign or security enforcement.

[0034] Except for the new terms that are defined below, the terms used in this document are consistent with terminology as defined in standard texts on Java, Enterprise Java Beans, WebLogic Server, and other generally accepted security concepts.

access control - the restriction of access to resources to prevent its unauthorized use.

access control information (ACI) - information about the initiator of a resource access request, used to make an access control enforcement decision.

access control list (ACL) - the list of entities, together with their access rights, that are authorized to have access to a resource.

authorization - the granting of authority, which includes the granting of access based on access rights.

credentials - information describing the security attributes (identity and/or privileges) of a user or other principal. Credentials are claimed through authentication or delegation, and used by access control.

delegation - the act whereby one user or principal authorizes another to use his (or hers or its) identity or privileges, perhaps with restrictions.

entitlement - a right or permission granted to a principal.

identity - a security attribute with the property of uniqueness; no two principals' identity may be identical. Principals may have several different kinds of identities, each unique.

principal - a user or programmatic entity with the ability to use the resources of a system.

role - an organizational identity that defines a set of allowable actions for an authorized user.

Role Based Access Control (RBAC) - a class of security mechanisms that mediate access to resources though organizational identities called roles.

security attributes - characteristics of a subject (user or principal) that form the basis of the system's policies governing that subject.

security policy - the data that defines what protection a system's security services must provide. There are many kinds of security policy, including access control policy, audit policy, etc.

Service Provider Interface (SPI) - a package or set of packages that support a concrete implementation of a subset of services.

target - the final recipient in a delegation "call chain". The only participant in a call-chain that is not an originator of a call.

target object - the recipient of a business request message.

trust model - a description of which components of the system and which entities outside the system must be trusted, and what they must be trusted for, if the system is to remain secure.

Trusted Computing Base (TCB) - the portion of a system that must function correctly in order for the system to remain secure. A TCB should preferably be tamper-proof and its enforcement of policy should not be able to be circumvented.

unauthenticated principal - a user or other principal who has not authenticated any identity or privilege.

user - a human being using the system to issue requests to objects in order to get them to perform functions in the system on their behalf

[0035] Embodiments of the invention described herein allow for the determination of entitlements as used within an access context. Typically, a user is trying to access a protected resource on the server. This protected resource may, for example, be a software application running on the server, a particular Web page or portion of a Web site, or a database system. Other types of protected resource may be used while

remaining within the spirit and scope of the invention. When the user attempts to access the resource, the security service determines the type of access request, the destination (the protected resource), and the context in which the request is made. From this information the security service can determine an entitlement, or a set of entitlements, for the user. Entitlements clearly denote what a particular user may do with a particular resource in a particular context. Entitlements can be used to represent the business logic or functionality required by the server provider, thus bridging the gap between a simple security platform, and a complex business policy platform.

[0036] Embodiments of the invention additionally allow for the integration of third party vendor security products to provide protection for such application types as:

- Enterprise Java Beans
- Web Applications (Servlets, Java Server Pages (JSP's))
- Resources (Remote Method Invocation (RMI), Java Messaging System (JMS))

[0037] Embodiments of the invention also allow for the integration of external public key infrastructure to support such advanced features as:

- Certificate and key retrieval
- Certificate validation
- Revocation mechanisms (Certificate Revolution List (CRL), Online Certificate Status Protocol (OCSP))

[0038] An important goal in developing the invention is to include support for the Java 2 Enterprise Edition (J2EE) specification and interoperability therewith. These J2EE specification features include the Common Secure Interoperability (CSI) protocol, user identity tokens, the Stateless Authentication Service (SAS) protocol, support for propagation

of security credentials across machine, cluster, and/or domain boundaries, control of propagation of identity based on policy, enhanced support for virtual host/sites, the ability to generate a user identity scoped to domain, and host/site specific security policies.

[0039] Another important goal of the invention is to include support for features such as updated SSL capabilities, hardware accelerators, Transport Level Security (TLS) protocol, session resumption, certificate authorities, protection of network connections, Web server proxy, server to server communications, encryption of session ID, and enhanced integration with directory servers.

Security Domains

[0040] As used in the context of the invention, a security realm or domain can span a single server, a plurality of servers, a cluster of servers, or a management domain. The invention allows for administrative control of identity propagation within, across, and between security domains. The identity of a client application or user can be propagated and scoped by the security domain. For example:

```
Principal-Identity@security-domain-name
```

[0041] When propagation is disabled, the initiating identity can be represented as being anonymous. For example:

```
<anonymous>@initiating-security-domain-name
```

Framework

[0042] The security architecture or framework provided by the invention is intended to address the requirements put forth and many other things. Focusing exclusively on the authorization requirements, the security architecture attempts to provide the following capabilities in

addition to a typical permission-based authorization:

Dynamic Role Association

[0043] Dynamic Role Association is a mechanism to allow a late binding of the association of roles to principals, that is capable of taking the context of the request into consideration during the determination of the roles for which the principal is entitled. Dynamic Role Association can be thought of as the late binding of principals to roles at runtime. This late binding occurs just prior to an authorization decision for a protected resource regardless of whether the principal-to-role association is statically defined or dynamically computed. Because of its placement in the invocation sequence, the result of any principal-to-role associations can be taken as identity as part of any authorization decision made as part of this request.

[0044] Unlike statically defined associations, the association of principals to roles can be computed dynamically at runtime. The computation of a dynamic role is able to access a number of pieces of information that make up the context of the request, including the identity of the target (if available), the values of the request's parameters, the values of profile attributes associated with the initiating principal, as well as potentially other information.

[0045] The context information is typically utilized as values of parameters in an expression that is to be evaluated by a rules or expression evaluation engine. Depending upon the functionality provided by the evaluation engine, it may also be possible to express the ability to call out to an external class that performs external calculations, such as the amount of miles flown by the principal, that are then used as parameter of the expression.

[0046] In addition to computing any roles that should be dynamically associated with the principals in the Subject, this same functionality is also responsible for associating any roles that were

statically defined either through the consumption of a J2EE defined Deployment Descriptor or by an administrator.

[0047] The result of this capability is in one embodiment, an updated JAAS Subject that contains any roles that the principals contained in the Subject were entitled to receive given the context and the target. These roles are then available to any authorization decisions on protected resources, as well as business container and application code. For example, a personalization server, application server, or Web server, could utilize standard methods to determine if a particular role is held by the subject as a means to personalize a web page. This same mechanism could be used by an Enterprise Java Bean or other application to determine whether to retrieve certain fields from a record in a database, without having knowledge of the business policies that determine whether access should be allowed, resulting in field-level authorization.

Parametric Authorization

[0048] Parametric Authorization is a mechanism that allows an authorization decision about a protected resource to be determined that is potentially based on the context of the request. Within the architecture provided by the invention, the scope of enforcement has been broadened to apply to all execution and resource containers. This is accomplished by having all execution and resource containers obtain authorization and role mapping services through an AccessController object provided by a Security Service. The AccessController utilizes the authorization services provided through the Service Provider Interfaces in order to provide the requested capabilities.

[0049] The Service Provider Interfaces that actually define the mechanisms used to provide Dynamic Role Association and Parametric Authorization are based on a delegated authorization design that results

in the point of enforcement being moved to the provider of the functionality, instead of within the application server itself.

[0050] In addition, the methods defined on the Service Provider Interfaces that comprise the authorization framework utilize an enforcement mechanism designed to support a capabilities-based authorization mechanism that more naturally aligns with the services provided by 3rd-party authorization vendors. The use of a capabilities based mechanism allows the delegated authorization approach to support both capabilities-based as well as permission-based implementations.

[0051] Traditional security mechanisms tend to be context-less since they are based solely on permissions granted a principal for a given resource. Therefore, the only types of authorization decisions that can be made are whether the principal has the necessary permissions to access the resource. These types of authorization decisions are more complicated since they don't tend to represent business decisions. As a result, additional security checks are often required in application code to compensate for this limitation. For example, using traditional methods one could evaluate whether a principal is allowed to transfer money to a specific account. However, it is not possible to take into account the amount of the transfer, the currency of both the source and destination accounts, and the day of the week.

[0052] In a parametric authorization based mechanism in accordance with the invention, the authorization decision is made using the context of the request, therefore the authorization decision more closely represent real business decisions. There is very little need for compensating security checks in the application itself. For example, using the invention in a real-world banking example it is possible for the authorization decision to take into consideration the amount of the transfer, the currency of both source and destination accounts, and the day of the week as part of processing the business policy that controls whether transfers are allowed, by whom, and under what circumstances.

Access to Context Information

[0053] While some other systems have tried to provide capabilities similar to the description of parametric authorization described above, these earlier systems and methods all require the caller to have previous knowledge of the parameters of the business policy being evaluated. The requirement for this knowledge presents virtually the same issue as that with making the authorization decision in application code- knowledge of some aspect of the business policies. Any time the business policy changes to require another piece of context information, the application must be modified and re-deployed.

[0054] Addressing the issue of providing context information without prior knowledge of the business policy is accomplished by using callbacks to the containers from the authorization provider. While changing the type or number of parameters of a request fundamentally changes the application requires the application be re-deployed, it is possible for the implementations of dynamic role association and parametric authorization providers to obtain access to the context information without requiring the application to have prior knowledge of aspects of the business policies.

[0055] The Service Provider Interfaces defined to work with the invention and support the authorization framework can utilize a standardized callback mechanism similar to the one defined in the Java Authentication and Authorization Service (JAAS). Utilizing a callback handler that is passed from the container to the implementation of the Service Provider Interfaces, the provider is capable of requesting specific context information be returned. When the container's callback handler is called, the container responds by populating the callbacks with the appropriate values. If the container does not understand the callback specified in the handler, it does not update the callback with a value. Neither the application, nor the container is aware of changes to business

policy since they are queried for context information that is driven by the evaluation of specific business policy expressions.

Functional Description

[0056] Implementations of an entitlement processing engine (entitlement engine) that could be used to satisfy the requirements of parametric authorization must support the following requirements:

[0057] An implementation should be able to specify that the value of a parameter from the business request, primary key, attribute from a Principal contained in the JAAS Subject that represents an authenticated user, and any potentially other information considered part of the context be expressed as part of a rule expression.

[0058] An implementation should be able to request parametric information from the caller on-demand, instead of requiring any parametric information that could be potentially used by the evaluation of rules to be passed at the beginning of rule evaluation. The mechanism for requesting information should be in the form of a callback similar to that defined for JAAS. The implementation might callback for each piece of information individually, request multiple pieces of information in a single callback, or a combination of both.

[0059] An implementation should be able to specify the name of a Java class that will be called as part of the evaluation of a rule. The value returned from the Java class will be used in the evaluation of the rule expression. This can be thought of as a macro facility where user provided code is called to compute a value that is then taken into consideration when evaluating the rule. For example in a travel booking example, a rule expression could specify a call to a user-supplied class that computes or looks up the number of miles a passenger represented by the Subject has flown to date.

[0060] The evaluating of a set of rules should result in a boolean value that indicates whether the rule was successfully satisfied or not.

[0061] The invention provides enterprise application developers with a framework architecture within which application implementations, particularly login, authorization, and auditing implementations, may be deployed or "plugged in" to allow the development of readily adaptable and customizable business functions and scenarios. These systems can serve to interpret the principal identity in the context of a particular scenario, and to tailor business functions to meet the interpreted identity.

[0062] The framework architecture provided by the invention particularly allows Java developers to make use of existing Java security features, but to greatly expand the use of these security features in a manner that is transparent to the developer (or the user), and that can allow the security provided to be distributed throughout the enterprise without a need for additional layering or translation of security processes.

[0063] Embodiments of the invention are particularly useful in the field of enterprise and server security and business workflow. Since the framework uses a pluggable architecture, it allows third-party vendors to provide security implementation plug-ins without a need for the core architecture to be modified. These plug-ins may include tools for digital certificate and key retrieval, certificate validation, and revocation mechanisms (e.g. CRL and OCSP). The pluggable framework architecture thus allows for the secure access control of enterprise resources, such as enterprise Java beans (EJB's), applications (including servlets and JSP's), and other networked resources (such as RMI, JNDI, and JMS destinations).

[0064] Additionally, the framework architecture of the invention is especially suited to providing a method of secure authorization and secure resource access throughout an enterprise-wide security domain. Each server with a domain may provide a different set of features, services and applications. The invention allows developers to deploy client applications which can make use of each server's particular services through a single sign-on mechanism that determines the client's

access privileges throughout the domain or between cooperating domains.

[0065] **Figure 2** shows an example of a security architecture in accordance with an embodiment of the invention. As shown therein, clients **202**, **204** (which may be either physical hardware clients or software applications) may attempt to access a secured service or resource **206**, such as a persistent directory server, via a transaction or application server **208**. An example of such a transaction server is the Weblogic Server product from BEA Systems Inc., San Jose, California, although the invention may be used with any other server product or equivalent system. Internet CORBA clients will typically attempt to make such an access through an Internet Inter-ORB Protocol (IIOP) request **212**. Web clients will typically attempt to make an access through a series of hypertext transfer protocol (http) requests **210**, either directly via a Web server **214**, or via a proxy plug-in **216** (in which case the proxy may also provide additional functionality, such as, for example, secure socket layer (SSL) encryption **218**). In any case, the connection attempt is received by the transaction server, often via an initial connection filter **220**, and is passed to the security service **222**. In accordance with the invention, the security service **222** is the focal point for security determination, including client and user level resource access, authorization, certification, privilege assessment and entitlement determination. Enterprise Java Beans (EJB's) **224**, Web applications (WebApp's) **226**, and other forms of applications may all use the security service through the use of containers. The security service handles calls from these containers to the protected resource, which in the case of Figure 2. The calls may be handled by, for example, a plurality of managed beans (MBeans) **230**, or the Java Named Directory Interface (JNDI) **232**.

[0066] **Figure 3** illustrates an embodiment of the security service architecture **300** in greater detail. The security service augments the

basic security services and features provided by the standard Java2 Enterprise Edition security set. As shown in this example, the basic Java security set **302** includes security provider interfaces **304** for key storage, authentication, certificate validation, and secure sockets, among others. Customer applications **306** may be written to directly take advantage of the Java security layer and these SPI's. The invention, as shown in **Figure 3** greatly enhances the developer's options in making use of these and other security features. In accordance with the invention, customer applications are deployed in containers, for example, an EJB container **308** of a WebApp container **310**. The containers communicate directly with the security service **314** (herein the same as security service **222**), which in turn communicates with the Java security layer **302** and its security SPI's **304**. This allows the responsibility for secure authorization decisions to be moved from the application and placed in the security service layer.

[0067] In addition, the invention is ideally suited to integrating the use of preexisting Java security SPI's **304** with custom SPI's **316, 318**. Additional SPI's such as, for example, a connection filter, an access decision interface, an audit channel, and a certificate authenticator, may be included or integrated with the security server. The security service mediates access to the entire range of secured enterprise resources.

[0068] **Figure 4** illustrates a process by which an application may use the invention to access a secure resource. In step **402**, a developer creates a customer application. The application may be one used by a physical user or customer, or may be an application interface program to allow another application to make use of the security server. In step **404**, the application is deployed within a container, the deployment process of which is disclosed in further detail below. The application then, in step **406**, accesses the security service using this container. Optionally, in step **408**, the security service can interface directly with a set of J2EE compliant Java security features. Alternatively, in step **410**, the

containers are used to allow access to a third-party or custom Security Provider Interface. In step **412**, the Security Provider Interface allows the application to access the secure resource, provided that the security service has indicated that the access requested should be allowed.

[0069] **Figure 5** illustrates one embodiment of the deployment mechanism used to deploy an application within a container. As shown in **Figure 5**, data is registered in the security service **502** (herein the same as security service **314**) for use by the application container **504**. Traditionally, the container reads the information from the deployment description **506**, and the actual container then does the enforcement. If there are different kinds of containers, one for WebApp, one for EJB, then there would need to be different implementation engines. In addition there exists a series of resources inside of the J2EE environment that do not have a container, and that don't have deployment descriptors. The primary problem with this method is that various aspects of security are enforced in different ways within a single environment. The invention addresses this problem by first allowing the container reading the deployment descriptor to inform the security service of the security constraints and policies defined within, resulting in the population of security policies within the security service. Secondly, the container delegates all authorization decisions to the security service when the container later makes a request to access a protected resource, thus moving the point of enforcement from the container to the security service. The authorization decision is made by consulting the security constraints and policies provided through the deployment description, as well as any defined by the administrator through an administrative tool. A deployment tool can also be used to read the deployment descriptor and record the security information contained within into the security service, The deployment tool can then go directly to the security service, without reading the deployment descriptor.

[0070] **Figure 6** illustrates a process by which data is entered into the security service during application deployment. In step **602**, the developer creates a deployment description. The description is passed to the container in step **604**, where the container analyzes the information to assess what information should be passed to the security service. Optionally, the deployment description could be given to a deployment tool, in step **606**, that passes the information to the security service.

[0071] **Figure 7** illustrates an example of a client using the security server in accordance with the invention to access a protected resource on the network or the server. The process starts with the client **702** making a request, which is one instance may be a simple case of the client authenticating itself. The client makes a connection to the container, indicated by the arrow **704**. A token **706** is passed along with the request. As a part of making the connection, the system (including the security server **710**) makes a determination that the resource **708** is protected, so the client needs to authenticate. In this example, tokens represent the authenticated users of the client that are in turn being passed along in the invocation request to the container that the container will then use in the later step of calling the security service. In some embodiments the token represents the authenticated identity of the client itself. The client poses to the security service **712** (the same security service or security service **314**) the question **714**, "Can the user identified by this token get access and perform the requested capabilities on the target resource?"

[0072] To answer this question the security service uses an Access Controller, indicated as AC **716** in **Figure 7**. The AC **716** does two things: it acts as a fanout spreading out through a series of Access Decision (AD) plug-ins **718**, **720**, **722**, and it also acts as an adjudicator. When the AC calls the Access Decision it passes the token along to each Access Decision in turn, beginning with the first one, Access Decision **718**, asking, "is this OK?" The Access Decision returns one of three

responses: permit, deny, or abstain. The Access Controller then moves on to the next one if there is a next one, and so on until all Access Decisions have been polled. In effect the security mechanism acts like a series of panes of glass. If X **718** says "permit" the process continues on; if X **718** says "deny" the process stops. The AC then moves to Access Decision Y **720** and asks the exact same question presenting the exact same identity, and Y **720** similarly responds with a permit, deny or abstain.

[0073] This process continues for however many of these plug-ins the service has. An adjudication policy determines that if anyone says deny, then the process terminates, or simply returns an overall deny. The architecture allows for the adjudication policies which make this decision to be flexible enough that if X **718** is permit, and Y **720** is deny, and C **722** is abstain, then that can be considered a permit. Other implementations can take a more hard line: if there's any denying, it's an overall denial. An abstain is returned whenever the AC asks the access decision a question that the access decision doesn't exclusively understand, and hence cannot make an explicit decision. This is most common whenever a legacy authorization system is included that is needed for some but not all operations, or when X **718** is corporate policy, and Y **720** is line of business. In this case there may be questions asked that are enforced by corporate policy but not line of business, so the line of business decision maker says, "I don't understand this, I abstain".

[0074] When all of the access decision makers are polled and no one up to the point where the request finally gets through says deny, then the access request may reach a point that nobody denies, some permit and some abstain, for example, where Y **720** abstains and X **718** and C **722** both permit. The adjudication policy can be then used to control how to determine the outcome. In one embodiment there are only two outcomes to the container: a permit or a deny. So the access controller looks at the adjudication policy to determine how that decision is made.

For example in some, out of the box implementations, the system may use an adjudication policy which says, "do I require unanimous permission?" In which case, X, Y and C all have to permit. If any one of them does not permit, then the outcome is a deny. The other approach is to not require unanimous consent, in which case as long as no one denies then the access is allowed to occur. This decision is made based on the adjudication policy.

[0075] No matter what happens within the access decision process the security service itself passes requests to the auditing subsystem **724** requesting that it audit what has just occurred and the outcome. Only at that point will the security service then tell the container that the final result is a permit or a deny. If it is permit, the container then allows the requested operation to be dispatched on to the protected resource. If it says deny, the request does not get to that point, and is immediately rejected.

[0076] The invention also provides a method to call the Access Decision process on the way out i.e., following a successful receipt of an access request at the protected resource. In this scenario it can be assumed the system went through the first checks on the request and everything was permitted, so the method was dispatched to the resource. The resource returns data that is intended to go out back to the client. Rather than just blindly returning this data back, the authorization check is rerun, this time allowing the security service, particularly the Access Decisions to look at the data to be returned to determine if its okay to return to the client. An example here is that of a business operation where a user is trying to request documents. When the user first enters they have a security clearance assigned to them, which means that they can look at documents that aren't classified above their security clearance. If they only have a secret security clearance and the documents that were to be returned have top secret, then even though they got the method to run, the system would not want to allow the

outcome to go back to the user because they would then be allowed to see something that they are not permitted to see. The Access Decisions are re-run again, this time reviewing the output that's going to go back to the user and make that same kind of decision. In this example there may be a lot of abstains, for example the Y Access Decision **720** says, "I don't care what the output is going back, I only care about the input coming in". The developer can mix and match plug-ins to achieve different policies based on line of business, or based on corporate policy.

[0077] **Figure 8** illustrates a flowchart showing the process used by an embodiment of the invention in providing access to a secured or protected resource. In step **802**, the client develops and deploys the application within a container. When the application requires access to the protected resource, a call is made and forwarded to the appropriate container shown by step **804**. The container is used to invoke the security service in step **806**. Within the security service, in step **808**, a fanout access controller determines which access decisions are to take place in verifying or acknowledging this particular access request. In step **810** each Access Decision is selected, and processed or polled to determine its contribution to the access request, the output of each access decision being a deny, permit, or abstain. An audit mechanism tracks the outcome of each Access Decision in step **812**. Based on the outcome of the access decision and the cumulative response -- be it permit, deny, or abstain -- in step **814** the container is given (or denied) access to the protected resource. In step **816** data can then be passed from the client application to the protected resource and vice versa. If the client later requests either to the same resource or a different resource (as determined in step **818**), then the request is forwarded to the appropriate container and another authorization decision is required (step **820**), otherwise the session terminates (step **822**).

[0078] **Figure 9** illustrates the security and personalization layers in accordance with one embodiment of the invention, including the

entitlements layer **904**, and its positioning in relation to the business application layer **908**, business policies layer **906** and security authorization layer **902**. It will be evident to one skilled in the art that the illustration shown in **Figure 9** is an abstract representation of how an entitlements layer may be developed, but that many other variations on this architecture can be used, and that such discrete layers might not (and in fact most likely would not) be used in any particular design implementation. **Figure 9** shows how entitlements **904** can be used to transcend the boundary between the traditional security architecture (represented as **902**), and the often-distinct business policy architecture (represented as **906**). The information garnished through the use of such entitlements is used to direct the operation of a company's business applications represented as **908**, including personalizing them for individual entities based on that entity's entitlement.

[0079] **Figure 10** shows how entitlements can be used to generate roles dynamically. An entitlement process **1002** allows a security question to be asked within a context. The invention includes an ability to dynamically assign roles to a particular subject **1000**, for example Principal 1 **1004**, Principal 2 **1006**, through Principal *n* **1008**, based on that entitlement, which further allows the system to provide an application context within the security layer, and allows authorization questions to be phrased in terms of the application. The system can only accurately answer those questions if they are presented in terms of the context, like the common example of a doctor and a patient. If the question is posed "Can Dr. Smith modify a patient's chart?", then it is a context-less question. The traditional answer is "sometimes yes and sometimes no". The only way to properly determine the answer is to know the context in which the question is asked, i.e., which patient are we talking about, and perhaps which chart. The system can then make role assignments like ownership or basically dynamic assignments that allows the system to make business decisions rather than security decisions.

[0080] Using a simple notation we can represent the concept as follows:

For each subject, an active role AR is one that the subject is currently using:

AR(s:subject)={active role for subject s}

Each subject may be associated with one or more authorized roles RA:

RA(s:subject)={authorized roles for subject s}

Each role may be authorized to perform one or more tasks TA:

TA(r:role)={tasks authorized for role r}

Subjects may execute tasks if the predicate function exec(s,t) is true at the current point in time with the current values:

exec(s:subject,t:task)=true iff subject s can execute task t

[0081] In order for the above statements to evaluate correctly, a couple of rules are required:

1. Role assignment - a subject can execute a transaction only if the subject has been assigned a role. The identification and authentication process are not considered part of the task, whereas all other activities performed on the system are conducted through tasks:

∀s:subject,t:task(exec(s,t)AR(s)⇒≠0)

2. Role authorization - a subject's active role must be authorized for the subject. This rule ensures that users can only take on roles for which they are authorized:

∀s:subject(AR(s)⊆RA(s))

3. Task authorization - a subject can execute a task only if the task is authorized for the subject's active role:

$\forall s:subject, t:task(exec(s,t) \Rightarrow t \in TA(AR(s)))$

[0082] As another example, a developer can use the invention to personalize Web pages based on what a user is entitled to do. For example, a self-registration or self-help feature can be made a part of a credit card website where, the user is the only one that can change their credit card numbers. Vendors can read them but they can't change them, and nobody else can see them. The system cannot figure out how to do this unless it knows the context. Can someone modify the credit card field? Of course someone can modify the credit card field. So, now, who can modify it? This is part of the context. Then the system must ask which particular one, because it can't just let any member modify any other member's credit card. So the system must know which object is under consideration, and which profile so it can then ask, "are you the owner of that profile?" In the past the only thing defined were static roles. What the invention does through the use of entitlement is allow, business analysts and business policy makers to define dynamic roles which are computed at run time in order to define to the object or the principal identity. These roles do not violate the security specification of the underlying server architecture. For example, a request may be a caller enrolled for the owner role. The system does not define the owner role, but it can have it dynamically associated to the user and then revoked from the user, as soon as they try to do something that they aren't the owner of anymore. Methodologies such as EJB can take advantage of this in that should the use be allowed to do something is based on whether or not you're in that role and you use the standard EJB to query that capability.

[0083] Another example of how the invention may be used is through the provision of a single sign-on mechanism, which may combine elements of a traditional security implementation with elements of a business policy implementation. As shown in **Figure 11**, information

received during an initial login process at a first security realm A **1102**, for example a security realm which is provided by a third-party application and has no knowledge of a user's profile, or of a corporation's business policies, can be captured by a first login module **1108**, and passed (indicated by the arrow **1114**) to a second login module **1110** in a second security realm **1104**. The second security realm B **1104** may have a knowledge of the business policies, and can determine or calculate entitlements or entitlement profiles **1102** for a user in addition to the original login information. The original login information, and the entitlement information (indicated by arrows **1118** and **1120** respectively) may be passed to a third login module **1112** in a third security realm C **1106**, and so on.

Security Provider Interfaces

[0084] Security Provider Interfaces (SPI) that can be used with the invention include interfaces for:

- Authentication – Java Authentication and Authorization Service (JAAS) LoginModule
- Authorization – AccessDecision
- Auditing – AuditChannel
- Principal Mapping->Role Mapping – JAAS LoginModule
- Principal->Credentials Mapping – JAAS LoginModule
- Key Storage – Java 2 KeyStoreSpi
- Certificate Retrieval and Revocation – CertPath
- Certificate Mapping – CertAuthenticator
- Hardware Accelerators – Java Cryptographic Extensions
- Perimeter Protection – ConnectionFilter

[0085] It will be evident to one skilled in the art that the SPI's described herein are merely examples of the type of interface that can be used with the invention, and are listed here for purposes of illustrating the

flexibility of the security service architecture provided by the invention. Other types of interface can be used to replace or augment those described, while remaining within the spirit and scope of the invention.

Authentication SPI

[0086] The JAAS Login Module used by the invention is based on the standard JAAS LoginModule, but additionally supports standard Login Module implementations. The authentication SPI is extended to support management of users, groups, and to provide support for multiple JAAS Login Modules. Credentials can be shared across Login Module instances, and a separate Login Module to update with user profile information can be provided. The responsibilities of the Authentication SPI include authentication of users based on security realm scope, and the population of the principal in the JAAS Subject.

Authorization SPI

[0087] The Authorization SPI provides for Access Decision, for which there is no Java equivalent sufficient. The Authorization SPI supports a variety of authorization models, both declarative and rules based. A callback handler can be used to obtain access to invocation context. The SPI also provides access to subject information, including principal identities, group membership and role assignments, and user profile information (optional). Multiple Access Decision providers can be configured in stack. The responsibilities of the Authorization SPI includes making authorization decision of permit, deny, or abstain, or access requests for protected resources scoped by application. **Figure 11** illustrates an example of how the invention can be used to distribute profile information between security realms within a domain.

Auditing SPI

[0088] The Auditing SPI includes an Audit Channel, for which no Java equivalent is defined. A callback handler is used to obtain access to invocation context including access to subject information, principal identities, group membership and role assignments, and user profile information (optional). Again, Multiple Audit Channel providers can be configured in stack. The responsibilities of the auditing SPI include determining if information should be audited, and performing the actual audit of the data based on QoS policies.

Principal->Role Mapping SPI

[0089] The Principal->Role Mapping (JAAS LoginModule) SPI is based on the JAAS Login Module, and is used to dynamically map roles to the identities of the principals contained in the Subject. The roles may have been explicitly defined by an administrator or from a deployment descriptor, or they may be dynamically computed based on the parameters fo the business request, the value of attributes in a principal's profile, as well as other conditions.

Principal->Credential Mapping SPI

[0090] The Principal->Credential Mapping (JAAS LoginModule) SPI is based on the JAAS Login Module, and is used to map principal identity when cross security domain policy or technology boundaries. The responsibilities of the Principal Mapping SPI is based on the Subject provided, and can be used to add public credentials with appropriate information to subject, such as password credential for username/password, and generic credential for token-type credentials

Key Storage SPI

[0091] The Java 2 KeyStore SPI is based on the Java 2 KeyStore interface, and provides consistent access to keys stored in a variety of

medias, including File-based: (PKCS #5/#8, PKCS #12), HSM-based: (PKCS #11), Smart Card/Java Card. The Key Storage SPI provides necessary support for code signing, and its responsibilities include retrieval of private key from protected storage, and retrieval of secrets from protected storage.

Certificate Retrieval SPI

[0092] The Certificate Retrieval SPI (CertPath) is based on JSR 55, and provides consistent mechanism to retrieve, verify, and revoke digital certificates, in addition to support for multiple certificate types. The responsibilities of the CertPath SPI include: CertStore: retrieval of certificates from store via query; CertPath: retrieval of chain of certificates utilizing CertStore; CertPathChecker: validates a certificate; and CertPathValidator: validate a chain of certificates.

Certificate Mapping SPI

[0093] The Certificate Mapping SPI (CertAuthenticator) for which no Java equivalent defined, provides an extensible means to allow mapping between digital certificates and principal identity within a security domain. It further provides verification and revocation checks performed prior to calling of CertAuthenticator

The responsibilities of the certificate mapping SPI are based on a verified client's digital certificate, map to username and realm of principal.

Hardware Accelerator SPI

[0094] The Hardware Accelerator SPI uses Java Cryptographic Extensions, and is based on Java 2 security architecture. The responsibilities include providing cryptographic services under control of the security service.

Perimeter Protection SPI

[0095] The Perimeter Protection SPI includes a Connection Filter to provide a built-in connection filter implementation. It is focused on ease of use, and is configurable from the console via Mbeans, allows continued support for user-written filters, and for cascading filters, each executed in turn. This minimizes conditions customers must handle.

Security Management SPI

[0096] The Security Management SPI Integrate third-party vendors consoles into the server console and allows security to redirect to servlets from these vendors. The security management SPI provides support for the full lifecycle of users, groups, roles, and authorization info.

Single Sign-on (SSO) SPI

[0097] The Single Sign-on SPI uses LoginModule to map from the initiating principal's identity to resources identity, and to provide secure storage of resource credentials.

[0098] The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to the practitioner skilled in the art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

Claims:

What is claimed is:

1. A security system for allowing a client to access a protected resource, comprising:

an application interface mechanism for receiving an access request from a client application to access a protected resource, and communicating said access request to a security service;

a security service for making a decision to permit or deny said access request; and,

a resource interface for communicating permitted access requests to said protected resource.

2. The security system of claim 1 wherein said application interface mechanism includes an application container for reading an application deployment description and registering said deployment description within the security service.

3. The security system of claim 2 wherein said application container is an Enterprise Java Beans container.

4. The security system of claim 2 wherein said application container is a WebApp container.

5. The security system of claim 1 wherein said security service includes a plurality of access decision mechanisms for defining an access policy and for determining a contributory decision to permit, deny, or abstain from said access request.

6. The security system of claim 5 wherein said security service further includes an access controller for transferring said access request to said plurality of access decision mechanisms, and for combining said contributory decisions into an overall decision by the security service to permit or deny said access request.

7. The security system of claim 5 wherein said access decisions represent a business function related access policy.

8. The security system of claim 5 wherein access decisions may be added to the security service to reflect changes in the access policy.

9. The security system of claim 5 wherein said access decision mechanisms are used to define an entitlement for said client to access said protected resource.

10. The security system of claim 5 wherein a deny or abstain by any one of said access decision mechanisms causes the security service to deny the access request.

11. The security system of claim 5 wherein an abstain by any one of said access decision mechanisms does not cause the security service to deny the access request.

12. The security system of claim 5 wherein said security service further includes an audit mechanism for auditing the determinations of said plurality of access requests.

13. The security system of claim 1 wherein said resource interface includes an interface mechanism to pass requests to or from a protected resource.

14. The security system of claim 13 wherein said interface mechanism includes a Java J2EE security interface.

15. The security system of claim 13 wherein said interface mechanism includes a security provider interface.

16. The security system of claim 13 wherein said interface mechanism is included as a plug-in in said resource interface.

17. The security system of claim 1 wherein the security service further makes a decision on whether to permit or deny a response to said access request from said protected resource to said client.

18. A method of allowing a client to access a protected resource, comprising:

receiving at an application interface mechanism an access request from a client application to access a protected resource and communicating said access request to a security service;

making a decision at said security service to permit or deny said access request; and,

communicating via a resource interface a permitted access request to said protected resource.

19. The method of claim 18 wherein said application interface mechanism includes an application container for reading an application deployment description and registering said deployment description within the security service.

20. The method of claim 19 wherein said application container is an Enterprise Java Beans container.

21. The method of claim 19 wherein said application container is a WebApp container.

22. The method of claim 18 further comprising:

defining an access policy via a plurality of access decision mechanisms within said security service; and,

determining at each access decision mechanism a contributory decision to permit, deny, or abstain from said access request.

23. The method of claim 22 further comprising:

transferring via an access controller said access request to said plurality of access decision mechanisms, and combining said contributory decisions into an overall decision by the security service to permit or deny said access request.

24. The method of claim 22 wherein said access decisions represent a business function related access policy.

25. The method of claim 22 wherein access decisions may be added to the security service to reflect changes in the access policy.

26. The method of claim 22 further comprising:

using said access decision mechanisms to define an entitlement for said client to access said protected resource.

27. The method of claim 22 wherein a deny or abstain by any one of said access decision mechanisms causes the security service to deny the access request.

28. The method of claim 22 wherein an abstain by any one of said access decision mechanisms does not cause the security service to deny the access request.

29. The method of claim 22 further comprising:

auditing via an audit mechanism the determinations of said plurality of access requests.

30. The method of claim 18 wherein said step of communicating via a resource interface includes passing requests via an interface mechanism to or from a protected resource.

31. The method of claim 30 wherein said interface mechanism includes a Java J2EE security interface.

32. The method of claim 30 wherein said interface mechanism includes a security provider interface.

33. The method of claim 30 wherein said interface mechanism is included as a plug-in in said resource interface.

34. The method of claim 18 further comprising:

making a decision on whether to permit or deny a response to said access request from said protected resource to said client.

35. A method for determining a user entitlement to access protected resources in a secure environment, comprising:

receiving an access request from a user application to access a protected resource

invoking a security service with said access request;

determining a user entitlement to access said protected resource;

making a decision at said security service based on said user entitlement to permit or deny said access request; and,

the steps of either

(a) communicating a permitted access request to said protected resource, or

(b) denying a denied access request to said protected resource.

36. The method of claim 35 wherein said entitlement determines the type of access available to the user of said protected resource.

37. The method of claim 36 wherein said type of access includes any of view, modify, delete, or copy, any part or all of said protected resource.

38. The method of claim 35 wherein information about said user entitlement can be communicated from a first security realm to a second security realm.

39. The method of claim 38 wherein additional information from a first security realm can be used to modify the user entitlement, prior to communicating said information about said user entitlement from said first security realm to said second security realm.

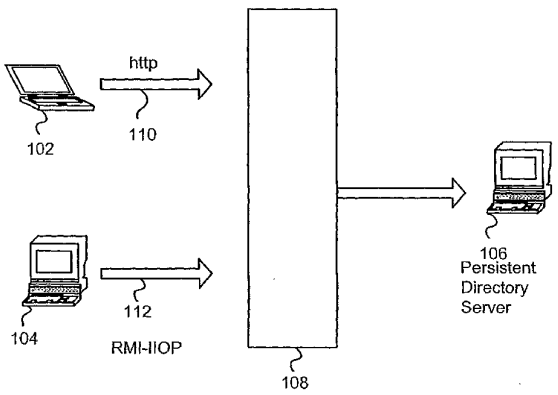

http
102
110
112
104
RMI-IIOP
108
106
Persistent
Directory
Server

FIGURE 1

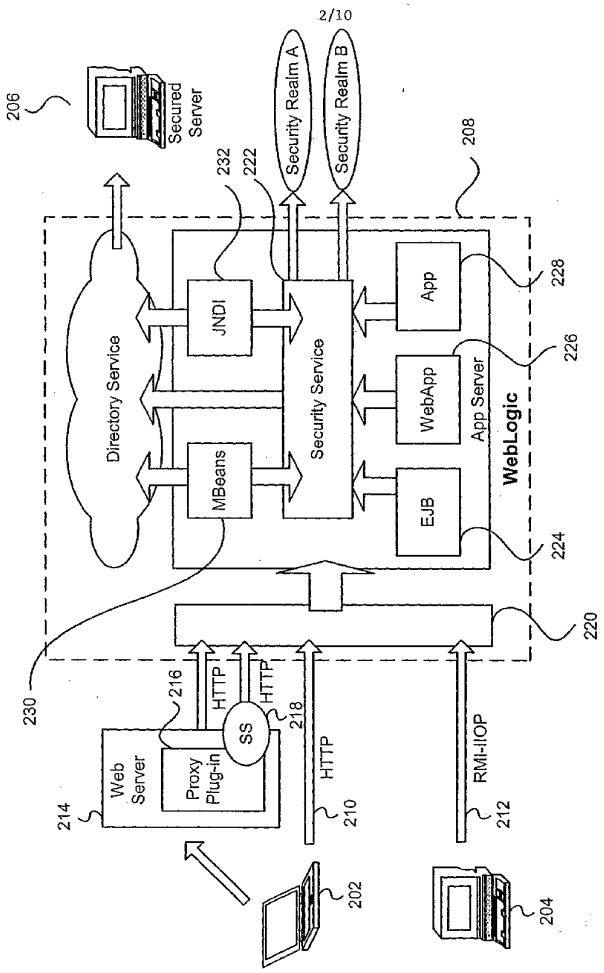
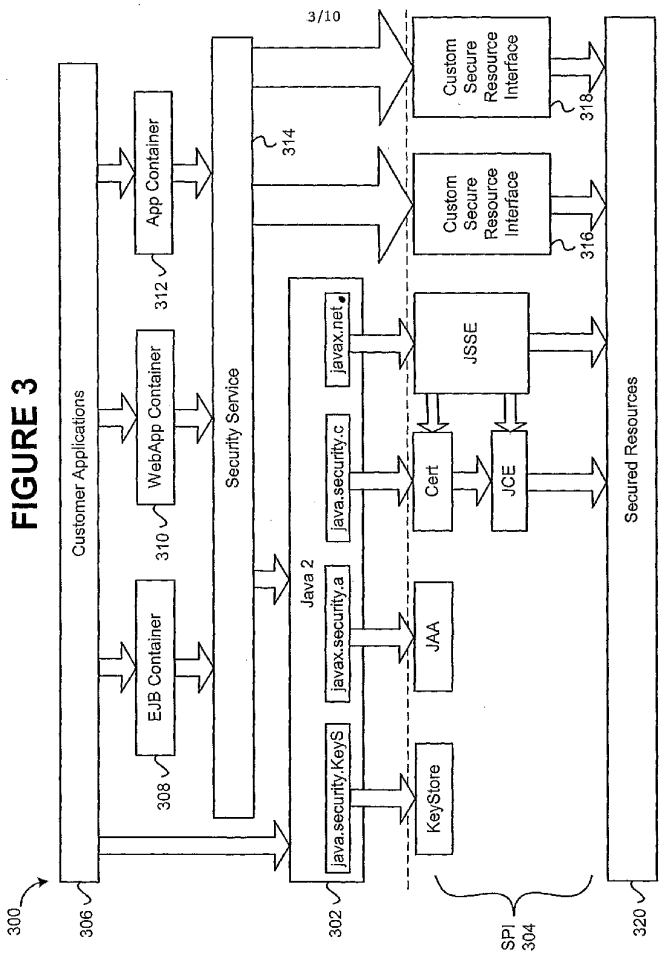

206
Secured Server
Security Realm A
Security Realm B
232
222
208
228
226
224
220
230
JNDI
Directory Service
MBeans
Security Service
App
WebApp
EJB
App Server
WebLogic
216
HTTP
HTTP
218
SS
Web Server
Proxy Plug-in
214
HTTP
210
RMI-IIOP
212
202
204

FIGURE 2

FIGURE 3


Customer Applications
App Container
WebApp Container
EJB Container
Security Service
Java 2
javax.net
java.security.c
javax.security.a
java.security.KeyS
JSSE
Cert
JCE
JAA
KeyStore
Custom Secure Resource Interface
Custom Secure Resource Interface
Secured Resources
SPI
300
306
312
310
308
314
302
304
318
316
320

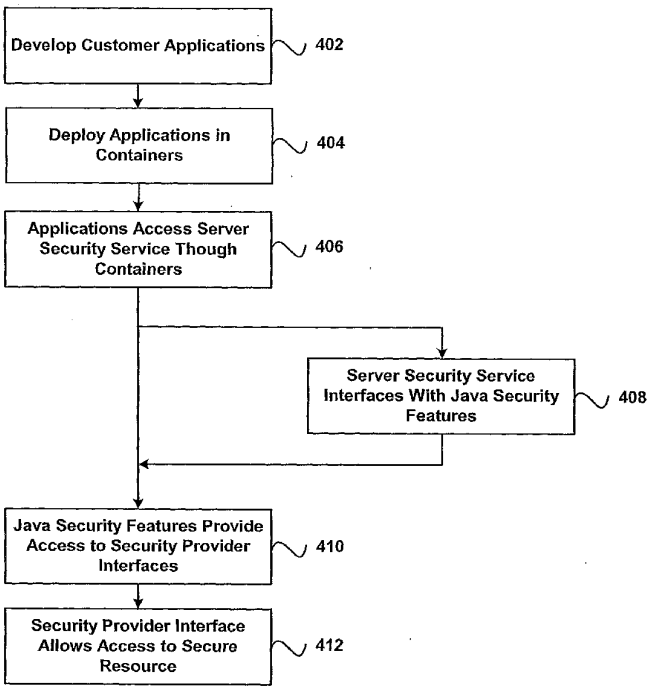

Develop Customer Applications
402
Deploy Applications in Containers
404
Applications Access Server Security Service Though Containers
406
Server Security Service Interfaces With Java Security Features
408
Java Security Features Provide Access to Security Provider Interfaces
410
Security Provider Interface Allows Access to Secure Resource
412

FIGURE 4

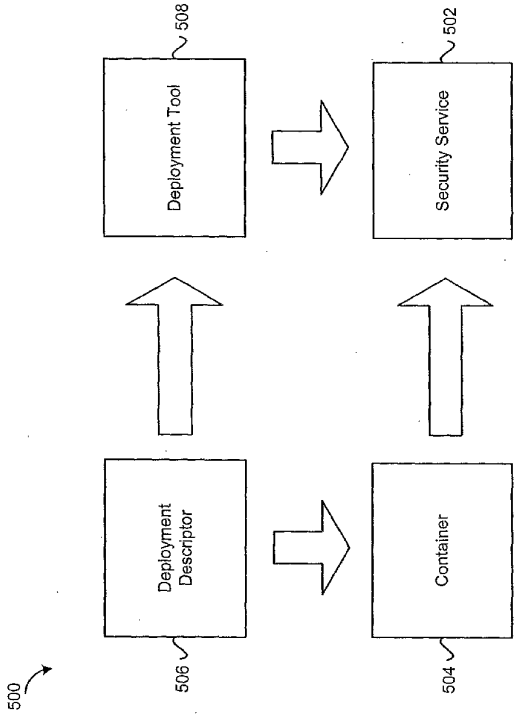

508
Deployment Tool
502
Security Service
Deployment Descriptor
Container
506
504
500

FIGURE 5

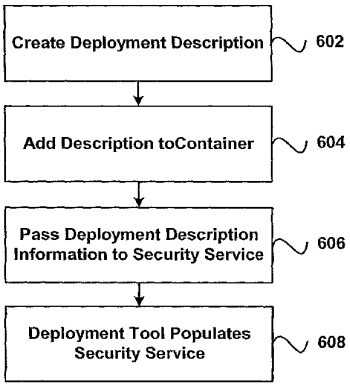

Create Deployment Description
602
Add Description toContainer
604
Pass Deployment Description Information to Security Service
606
Deployment Tool Populates Security Service
608

FIGURE 6

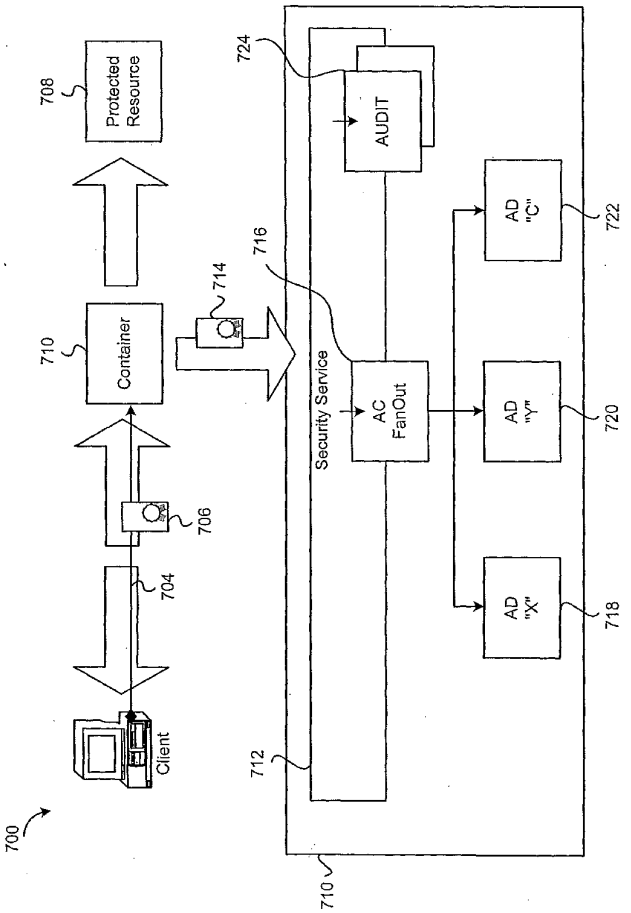

700
Client
704
706
Container
710
714
Protected Resource
708
716
Security Service
AC FanOut
AUDIT
724
712
710
AD "X"
718
AD "Y"
720
AD "C"
722

FIGURE 7

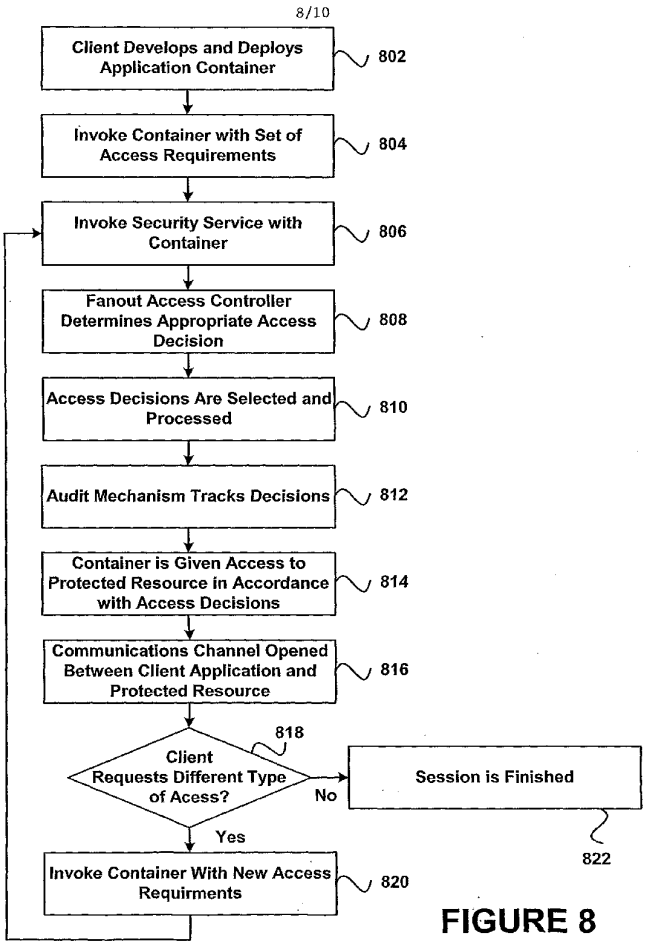

Client Develops and Deploys Application Container
802
Invoke Container with Set of Access Requirements
804
Invoke Security Service with Container
806
Fanout Access Controller Determines Appropriate Access Decision
808
Access Decisions Are Selected and Processed
810
Audit Mechanism Tracks Decisions
812
Container is Given Access to Protected Resource in Accordance with Access Decisions
814
Communications Channel Opened Between Client Application and Protected Resource
816
818
Client Requests Different Type of Acess?
No
Session is Finished
822
Yes
Invoke Container With New Access Requirments
820

FIGURE 8

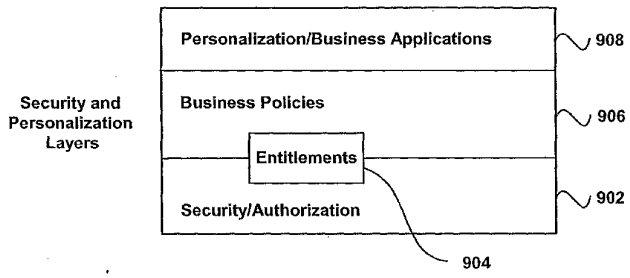


FIGURE 9

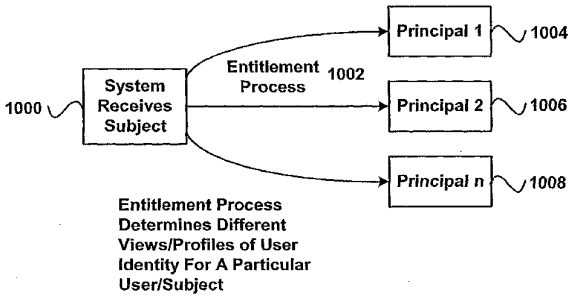


FIGURE 10

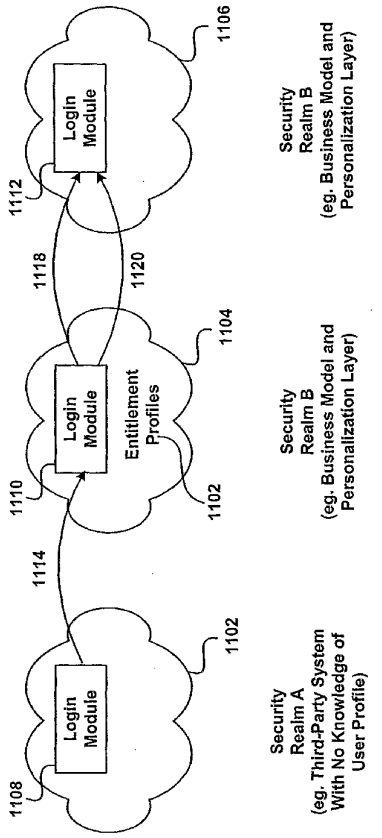

Login Module
1106
1112
1118
1120
1104
Entitlement Profiles
1110
1102
1114
1102
1108
Security Realm A (eg. Third-Party System With No Knowledge of User Profile)
Security Realm B (eg. Business Model and Personalization Layer)
Security Realm B (eg. Business Model and Personalization Layer)

FIGURE 11

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/16644

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : H04L 9/00
US CL : 713/176, 200
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 713/176, 200

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EAST,WEST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,925,126 A (HSIEH) 20 July 1999 (20.07.1999), fig.1 and 5-7; col.3, lines 43-67; col.4, lines 1-33;col.5, lines 45-67; col.6 and 8-9.	1-39

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 August 2002 (20.08.2002)	08/20/02 16 SEP 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Gilberto Barron Telephone No. 703-305-3900

Form PCT/ISA/210 (second sheet) (July 1998)

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,OM,PH,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TN,TR,TT,TZ,UA,UG,UZ,VN,YU,ZA,ZM,ZW

(74)代理人 100074228
弁理士 今城 俊夫
(74)代理人 100086771
弁理士 西島 孝喜
(72)発明者 パトリック ポール
アメリカ合衆国 ニューハンプシャー州 ０３１０９ マンチェスター コーブルストーン レイン ９

Ｆターム(参考) 5B085 AA08 AE23 BG01 BG02 BG07