(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5462254号
(P5462254)

(45)発行日　平成26年4月2日(2014.4.2)　(24)登録日　平成26年1月24日(2014.1.24)

(51)Int.Cl.		F I		
G06F 21/62	(2013.01)	G06F	21/24	163D
G06F 21/33	(2013.01)	G06F	21/20	133
G06F 21/44	(2013.01)	G06F	21/20	144B
G06F 21/57	(2013.01)	G06F	21/00	157A

請求項の数 18　(全 26 頁)

(21)出願番号	特願2011-516585 (P2011-516585)
(86)(22)出願日	平成21年6月24日 (2009.6.24)
(65)公表番号	特表2011-526387 (P2011-526387A)
(43)公表日	平成23年10月6日 (2011.10.6)
(86)国際出願番号	PCT/US2009/048461
(87)国際公開番号	WO2009/158405
(87)国際公開日	平成21年12月30日 (2009.12.30)
審査請求日	平成24年5月10日 (2012.5.10)
(31)優先権主張番号	12/163,164
(32)優先日	平成20年6月27日 (2008.6.27)
(33)優先権主張国	米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100080137
弁理士　千葉　昭男
(74)代理人　100096013
弁理士　富田　博行
(74)代理人　100147991
弁理士　鳥居　健一

最終頁に続く

(54)【発明の名称】コンピューティングプロセスのための最小特権アクセスの付与

(57)【特許請求の範囲】

【請求項１】

実行可能命令を含むコンピューター記憶媒体であって、該実行可能命令は、プロセッサーによって実行されると、

前記プロセッサー上で動作するオペレーティングシステムを用意するステップであって、前記プロセッサーは前記オペレーティングシステムを介してアプリケーションを実行する、ステップと、

前記プロセッサーが、前記アプリケーションによりアクセスされるリソース及び前記アプリケーションがなすことができるコールを制御するために前記オペレーティングシステムを介してセキュリティインフラストラクチャーを実施するステップであって、前記プロセッサーは、

前記プロセッサーがランタイムオブジェクトを作成する場合、前記プロセッサーが、関連付けられるサブジェクトのための一意の識別子を定義し、該関連付けられるサブジェクトに関連付けられるアカウントを指定するためのセキュリティ識別子を設けるステップ、

前記一意の識別子をメモリーに保持されるアカウントデータベース内のエントリにマッピングするステップであって、前記セキュリティ識別子に割り当てられた基本特権及び拡張特権を求める、マッピングするステップ、

前記プロセッサーが、前記関連付けられるサブジェクトのマイグレーションに沿って関連付けられるサブジェクトの識別子を蓄積するステップ、

前記プロセッサーが、前記セキュリティ識別子に割り当てられた前記基本特権及び前記拡張特権を求めることに基づいて、識別情報のセット及び該識別情報のセットに割り当てられた特権の集合を定義するためのセキュリティトークンを作成するステップ、並びに

前記プロセッサーが、

前記関連付けられるサブジェクトの前記蓄積された識別子のすべて及び前記特権の集合をインターセクトするステップ、並びに

前記リソースへのアクセスを付与する前に、前記関連付けられるサブジェクトのコールチェーンコンテキストをキャプチャするステップと、前記関連付けられるサブジェクトを分析して、前記コールチェーンにおける蓄積された各前記識別情報及び各前記関連付けられるサブジェクトが、前記要求されたリソースへのアクセスを有することを検証し、それによって、デフォルトにより最小特権を可能にするステップであって、前記コールチェーンはレジストリを開くためのレジストリオープンキーに対するアプリケーションコールを含み、前記レジストリは低い特権によって読み出すことができない保護されたキーを含み、前記アプリケーションコールは、前記レジストリのアクセスキーを携行しレジストリアプリケーションプログラムインターフェース（ＡＰＩ）を実施するサーバープロセスへマイグレーションされる、ステップ

によって前記関連付けられるサブジェクトのアクセス特権を求めるステップ

によって、前記セキュリティインフラストラクチャーを実施する、ステップと

によって、関連付けられるサブジェクトによるリソースへのアクセスを制御するためのセキュリティインフラストラクチャーを提供する、実行可能命令を含むコンピューター記憶媒体。

【請求項２】

前記関連付けられるサブジェクトは、プロセス又はプロセスのスレッドを含む、請求項１に記載のコンピューター記憶媒体。

【請求項３】

前記セキュリティトークンとセキュリティ記述子との間の関係が、要求されたリソースへの関連付けられるサブジェクトによるアクセス権の制御を決定する、請求項１に記載のコンピューター記憶媒体。

【請求項４】

アクセス特権を求める前記ステップは、セキュリティトークンにおける各前記セキュリティ識別子を使用して、セキュリティトークンが、セキュリティ記述子によって指定されたリソースへのアクセスを要求したか否かを判断し、コンタクトチェーン内のあらゆるコーラーが、要求されたリソースへのアクセスを付与する特権を有するときにのみ、前記要求されたリソースへのアクセスが付与される、請求項１に記載のコンピューター記憶媒体。

【請求項５】

前記リソースへのアクセス権を定義するセキュリティ記述子を構築するステップと、

前記セキュリティトークンに従って前記関連付けられるサブジェクトを識別するステップと、

前記セキュリティ記述子において定義された前記識別情報が前記セキュリティ記述子におけるアクセス制御エントリに含まれるか否かを判断するステップと、

前記アクセス制御エントリに基づいて前記識別情報にとって利用可能なアクセス権を求めるステップと、

前記求めたアクセス権に従って前記識別情報にアクセス権を付与するステップと

をさらに含む、請求項１に記載のコンピューター記憶媒体。

【請求項６】

関連付けられるサブジェクトに関係するすべてのトークンのリストをセキュリティトークンリスト内に保持するステップをさらに含み、現在のアクティブトークンは、常に前記セキュリティトークンリストの先頭にあり、前記関連付けられるサブジェクトに対するすべてのアクセスチェック及び特権チェックは、関連付けられるサブジェクトがアクセスを

要求するリソースを前記セキュリティトークンリストの先頭の前記現在のアクティブセキュリティトークンと比較することにより、前記現在のアクティブセキュリティトークン及び前記関連付けられるサブジェクトの現在のコンテキストを使用してハンドリングされる、請求項１に記載のコンピューター記憶媒体。

【請求項７】

クライアント側において、関連付けられるサブジェクトのセキュリティコンテキストをメッセージキューに書き込むステップと、前記サーバー側において、前記メッセージキューから前記関連付けられるサブジェクトの前記セキュリティコンテキストを非同期にリトリーブするステップと、及び偽装される前記関連付けられるサブジェクトの前記リトリーブしたセキュリティコンテキストをコピーすることによって前記関連付けられるサブジェクトを偽装するステップとをさらに含む、請求項１に記載のコンピューター記憶媒体。

【請求項８】

関連付けられるサブジェクトの現在のセキュリティトークンに関連付けられるすべての識別情報が要求リソースにアクセスすることができるときに、前記要求されたリソースへのアクセスを付与するステップをさらに含み、現在の関連付けられるサブジェクトのコールスタックにおけるすべてのチャンバーに関連付けられるすべての識別情報は、前記要求リソースにアクセスすることができ、前記関連付けられるサブジェクトのための保存されたコンテキストに関連付けられるすべての識別情報は、前記要求されたリソースにアクセスすることができる、請求項１に記載のコンピューター記憶媒体。

【請求項９】

オフライン処理用に、オフラインデータベース内にセキュリティコンタクトを記憶するステップをさらに含む、請求項１に記載のコンピューター記憶媒体。

【請求項１０】

プロセッサーによってオペレーティングシステムを介して実施されるセキュリティインフラストラクチャーであって、

関連付けられるサブジェクトのための一意の識別子を定義し、該関連付けられるサブジェクトに関連付けられるアカウントを指定するための、前記プロセッサーによって作成されるセキュリティ識別子と、

前記セキュリティ識別子に割り当てられた基本特権及び拡張特権を求めることに基づいて、識別情報のセット及び該識別情報のセットに割り当てられた特権の集合を定義するための、前記プロセッサーによって作成されるセキュリティトークンであって、前記基本特権及び前記拡張特権は、前記関連付けられるサブジェクトの１つ又は複数のアクセス特権を含み、前記１つ又は複数のアクセス特権は、

前記識別情報のセット及び前記特権の集合をインターセクトするステップ、並びに

前記リソースへのアクセスを付与する前に、前記関連付けられるサブジェクトのコールチェーンコンテキストをキャプチャするステップと、前記関連付けられるサブジェクトを分析して、前記コールチェーンにおける識別情報及び前記関連付けられるサブジェクトが、要求されたリソースへのアクセスを有することを検証し、それによって、デフォルトにより最小特権を可能にするステップであって、前記コールチェーンはレジストリを開くためのレジストリオープンキーに対するアプリケーションコールを含み、前記レジストリは低い特権によって読み出すことができない保護されたキーを含み、前記アプリケーションコールは、前記レジストリのアクセスキーを携行しレジストリアプリケーションプログラムインターフェース（ＡＰＩ）を実施するサーバープロセスへマイグレーションされる、ステップ

によって求められる、セキュリティトークンと、

要求されたリソースにアクセスすることができるアカウント及び前記プロセスに関するルールを定義するための、前記プロセッサーによって作成されるセキュリティ記述子と、

前記プロセッサーによって作成されるアクセス制御リストであって、該アクセス制御リストは、セキュリティ識別子のアクセス権を識別するための少なくとも１つのアクセス制御エントリを含む、アクセス制御リストと

を備える、セキュリティインフラストラクチャー。
【請求項１１】
前記セキュリティトークンは、構造体のバージョン、フラグ、オフセット、直接グループの個数、及びグループ識別子の総数を識別するためのフィールドを含む、請求項１０に記載のセキュリティインフラストラクチャー。
【請求項１２】
前記セキュリティトークンは、プライマリオーナーセキュリティ識別子、グループセキュリティ識別子、基本特権、及び拡張特権をさらに含む、請求項１１に記載のセキュリティインフラストラクチャー。
【請求項１３】
前記プライマリセキュリティ識別子及び前記グループセキュリティ識別子は、前記セキュリティトークンの関連付けられるサブジェクトの識別情報を定義する、請求項１２に記載のセキュリティインフラストラクチャー。
【請求項１４】
前記基本特権は、前記セキュリティトークンにおいて指定された前記識別情報に有効な特権のセットを含む、請求項１２に記載のセキュリティインフラストラクチャー。
【請求項１５】
前記拡張特権は、前記セキュリティトークンにおいてセキュリティ識別子について定義されたカスタム特権を含む、請求項１２に記載のセキュリティインフラストラクチャー。
【請求項１６】
関連付けられるサブジェクトに最小特権アクセスを付与するための、プロセッサーによって実行される方法であって、
プロセッサー上で動作するオペレーティングシステムを用意するステップであって、前記プロセッサーは前記オペレーティングシステムを介してアプリケーションを実行する、ステップと、
前記プロセッサーが、前記アプリケーションによりアクセスされるリソース及び前記アプリケーションがなすことができるコールを制御するために前記オペレーティングシステムを介してセキュリティインフラストラクチャーを実施するステップであって、前記プロセッサーは、
前記プロセッサーがランタイムオブジェクトを作成する場合、前記プロセッサーが、関連付けられるサブジェクトのための一意の識別子を定義し、該関連付けられるサブジェクトに関連付けられるアカウントを指定するためのセキュリティ識別子を設けるステップ、
前記一意の識別子をメモリーに保持されるアカウントデータベース内のエントリにマッピングするステップであって、前記セキュリティ識別子に割り当てられた基本特権及び拡張特権を求める、マッピングするステップ、
前記プロセッサーが、前記関連付けられるサブジェクトのマイグレーションに沿って関連付けられるサブジェクトの識別子を蓄積するステップ、
前記プロセッサーが、前記セキュリティ識別子に割り当てられた前記基本特権及び前記拡張特権を求めることに基づいて、識別情報のセット及び該識別情報のセットに割り当てられた特権の集合を定義するためのセキュリティトークンを作成するステップ、並びに
前記プロセッサーが、
前記関連付けられるサブジェクトの前記蓄積された識別子のすべて及び前記特権の集合をインターセクトするステップ、並びに
前記リソースへのアクセスを付与する前に、前記関連付けられるサブジェクトのコールチェーンコンテキストをキャプチャするステップと、前記関連付けられるサブジェクトを分析して、前記コールチェーンにおける蓄積された各前記識別情報及び各前記関連付けられるサブジェクトが、前記要求されたリソースへのアクセスを有することを検証し、それによって、デフォルトにより最小特権を可能にするステップであって、前記コールチェーンはレジストリを開くためのレジストリオープンキーに対するアプリケーションコー

ルを含み、前記レジストリは低い特権によって読み出すことができない保護されたキーを含み、前記アプリケーションコールは、前記レジストリのアクセスキーを携行しレジストリアプリケーションプログラムインターフェース（ＡＰＩ）を実施するサーバープロセスへマイグレーションされる、ステップ

によって前記関連付けられるサブジェクトのアクセス特権を求めるステップ

によって前記セキュリティインフラストラクチャーを実施する、ステップと

を含む方法。

【請求項１７】

クライアント側において、関連付けられるサブジェクトのセキュリティコンテキストをメッセージキューに書き込むステップと、

前記サーバー側において、前記メッセージキューから前記関連付けられるサブジェクトの前記セキュリティコンテキストを非同期にリトリーブするステップと、

偽装される前記関連付けられるサブジェクトの前記リトリーブしたセキュリティコンテキストをコピーすることによって前記関連付けられるサブジェクトを偽装するステップと、

前記コピーしたセキュリティコンテキストを分析するステップであって、前記要求リソースへのアクセス権を定義するセキュリティ記述子を構築する、分析するステップと、

前記セキュリティトークンに従って前記関連付けられるサブジェクトを識別するステップと、

前記セキュリティ記述子に定義された前記識別情報が該セキュリティ記述子のアクセス制御エントリに含まれるか否かを判断するステップと、

前記アクセス制御エントリに基づいて前記識別情報に利用可能なアクセス権を求めるステップと

をさらに含む、請求項１６に記載の方法。

【請求項１８】

関連付けられるサブジェクトの現在のセキュリティトークンに関連付けられるすべての識別情報が要求リソースにアクセスすることができるときに、前記要求されたリソースへのアクセスを付与するステップをさらに含み、前記関連付けられるサブジェクトのコールスタックのすべてのチャンバーに関連付けられるすべての識別情報は、前記要求リソースにアクセスすることができ、前記関連付けられるサブジェクトの保存されたコンテキストに関連付けられるすべての識別情報は、前記要求されたリソースにアクセスすることができる、請求項１６に記載の方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、コンピューティングプロセスのための最小特権アクセスの付与に関する。

【背景技術】

【０００２】

[0001]デバイスは、リソース、アプリケーション等へのアクセスを制御するセキュリティインフラストラクチャーで構成される場合がある。現在、移動デバイスでは、トラスト（trust）がコード識別情報に基づいて個々のアプリケーションに割り当てられる。セキュリティインフラストラクチャーは、どのようなアプリケーションがデバイス上で動作し得るのか、どのようなアプリケーションをロックアウトすることができるのか、どのようなアプリケーションをどのようなコンテキストで動作させることができるのか、及びこのようなアプリケーションがどのようなリソースにアクセスすることができるのかを判断する。現在のトラストレベルは、未署名のモジュールについて「トラステッド（信頼できる、trusted）」、「ノーマル」、及び「アントラステッド（信頼できない、untrusted）」を含む。アプリケーション又はモジュールは、トラステッド証明書、又はトラステッド証明書ストアにチェーンする証明書で署名を受けることができる。この場合、そのモジュールはトラステッドであるとみなされ、したがって、そのモジュールのどのコードも、シス

テム上のすべての特権を有するＡＰＩ及びリソースにアクセスすることができる。トラスト判定は、コーラー（caller）のアプリケーショントラストレベルに基づく。

【発明の概要】

【発明が解決しようとする課題】

【０００３】

[0002]しかしながら、上述した現在のセキュリティモデルにはいくつかの問題がある。スレッドがシステムの複数の保護されたサーバーライブラリ（ＰＳＬ）サーバーを通じてマイグレーションするとき、コールチェーンの直接のコール側プロセスがシステムリソースへのアクセスに対するパーミッションを有することが可能であるが、コールチェーンコンテキスト全体を検査することによって、リソースがアクセス可能であるべきでないことが明らかにされる場合がある。このようなセキュリティは、セキュリティ判定が常に直接のコーラーのコンテキストにのみ基づいている場合に強制することができない。また、現在のセキュリティモデルは、低い特権を有するアプリケーションがシステムサービスをコールする場合において異なる識別情報に偽装することにも備えておらず、システムサービスは、コーラーのコンテキスト又はそれ自身のコンテキストのいずれかに基づいて要求を処理するように求められる。またさらに、非同期アクセス要求について、現在のセキュリティモデルは、コーラーのコンテキストが完全に利用可能でないことがあり得る場合において２次スレッドのセキュリティチェックに備えていない。

【０００４】

[0003]本出願は、本発明が行ったこれらの検討事項及び他の検討事項に関するものである。

【課題を解決するための手段】

【０００５】

[0004]この概要は、詳細な説明でさらに後述する概念のうちの選択したものを簡略化した形態で紹介するために設けられている。この概要は、特許請求される主題の重要な特徴も本質的な特徴も特定するように意図されたものではなく、また、特許請求される主題の範囲を決定することを助けるものとして意図されたものでもない。

【０００６】

[0005]デフォルトによる最小特権を可能にするために、現在のスレッドの識別情報及び現在のスレッドのコールチェーンコンテキストの双方を考慮することによってセキュリティ判定を行う実施の形態が提供される。現在のスレッドコンテキストがキャプチャされ、そのコピーが作成されて、セキュリティチェックを非同期に実行するのに使用される。システムにおけるあらゆるスレッドが、関連付けられる識別情報を有する。最初に、この識別情報は、親プロセスから導出される。しかしながら、スレッドの寿命の間、この識別情報はスレッドの任意の偽装に基づいて変化し得ることが可能である。

【０００７】

[0006]これらの特徴及び利点並びに他の特徴及び利点は、以下の詳細な説明を読み、関連付けられる図面をレビューすることによって明らかになるであろう。上述の一般的な説明も以下の詳細な説明も、例示にすぎず、特許請求される本発明を限定するものではないことが理解されるべきである。

【図面の簡単な説明】

【０００８】

【図１】[0007]本発明の一実施形態によるセキュリティ識別子（ＳＩＤ）１００を示す図である。

【図２】[0008]本発明の一実施形態によるセキュリティトークン構造体２００のレイアウトを示す図である。

【図３】[0009]本発明の一実施形態によるセキュリティ記述子のレイアウト３００の簡略化された図である。

【図４】[0010]本発明の一実施形態によるアクセス制御リスト（ＡＣＬ）４００を示す図である。

【図５】[0011]本発明の一実施形態によるアクセス制御エントリ（ＡＣＥ）５００のための構造体を示す図である。
【図６】[0012]本発明の一実施形態によるセキュリティ記述子（ＳＤ）６００のための構造レイアウトを示す図である。
【図７】[0013]本発明の一実施形態によるスレッド７００のトークンリストのためのアレンジメントを示す図である。
【図８】[0014]本発明の一実施形態によるＰＳＬコール８００におけるスレッドのためのコールスタックリストを示す図である。
【図９】[0015]本発明の一実施形態によるスレッド９００のための、偽装リストとコールスタックリストとの間のリンク９００を示すブロック図である。
【図１０】[0016]本発明の一実施形態によるメッセージキューシステム１０００を示す図である。
【図１１】[0017]本発明の実施形態を実施することができるコンピューティング環境を示す図である。
【発明を実施するための形態】
【０００９】
[0018]実施形態は、既存のオペレーティングシステムの上部で動作して、どのようなリソースがアプリケーションによってアクセス可能であるのか及びどのようなＡＰＩをアプリケーションがコールすることができるのかを制御するように構成することができるセキュリティインフラストラクチャーを提供する。デフォルトにより最小特権を可能にするために、現在のスレッドの識別情報及び現在のスレッドのコールチェーンコンテキストの双方を考慮することにより、セキュリティ判定が行われる。現在のスレッドコンテキストがキャプチャされ、そのコピーが作成されて、セキュリティチェックを非同期に実行するのに使用される。システムにおけるあらゆるスレッドは、関連付けられる識別情報を有する、最初に、この識別情報は、親プロセスから導出されるが、スレッドの寿命の間、この識別情報は、スレッドが行う任意の偽装に基づいて変化し得ることが可能である。
【００１０】
[0019]実施形態は、どのようなコール又はアプリケーションが要求しているのかを分析するだけでなく、そのコールがどこから生じたものであるのかも分析する。実施形態は、エンドユーザー又はエンドオペレーターが自身の制御下にない場合がある異なるアプリケーションをインストールすることができるようにオープンエンドであるシステムのためのセキュリティインフラストラクチャーも提供することができる。したがって、スレッドがアクセスする権限を有しないリソースから該スレッドが誤ってアクセスされることに基づいてセキュリティ問題が発生することを防止するために、システムにおけるすべてのスレッドは、デフォルトにより最小特権で動作することができる。それどころか、特定のリソースにアクセスするときは、各コーラー及び各スレッドがそのリソースにアクセスすることができることを確認するために、現在のスレッドのすべてのコーラーが分析される。各コーラー及び各スレッドがそのリソースへアクセスすることができるときにのみ、そのコーラーには、そのリソースへのアクセスが与えられる。
【００１１】
[0020]本発明の実施形態は、どのリソースがアプリケーションによりアクセス可能であるのか及びアプリケーションがどのようなＡＰＩをコールすることができるのかを制御するためのセキュリティインフラストラクチャーを提供するデータ構造体を使用して実施することができる。このデータ構造体は、次の要素、すなわち、セキュリティ識別子（ＳＩＤ）、セキュリティトークン、セキュリティ記述子（ＳＤ）、アクセス制御リスト（ＡＣＬ），及びアクセス制御エントリ（ＡＣＥ）を含む。
【００１２】
[0021]図１は、本発明の一実施形態によるセキュリティ識別子（ＳＩＤ）１００を示す。ＳＩＤ１００は、システム内で一意のＩＤを定義し、特定のスレッドがどのようなアカウントで動作しているのかを指定する可変長データ構造体１１０である。ＳＩＤ１００は

、ユーザーアカウントのようなものであると考えることもできる。例えば、或る人がコンピューターにログオンしたとき、さまざまなユーザーアカウントが、適切なログオンパスワードを使用してアクセス可能になる場合がある。ユーザーアカウントは、コンピューターにログオンしている人を識別する。ＳＩＤ１００は、単にスレッド又はプロセスを識別するアカウントである。ＳＩＤ１００は、特定のデバイス内においてのみ一意である。しかしながら、当業者は、本明細書における本発明の実施形態の説明を分析した後、グローバルセキュリティ識別子を使用することができることを認識する。

【００１３】

[0022]デスクトップ上において、ＳＩＤ１００を、個々のユーザーアカウント、一定のシステムアカウント、及び一定のユーザーグループに割り当てることができる。ＷＩＮＤＯＷＳ（登録商標）ＣＥオペレーティングシステムにおけるアカウントは、例えば、デスクトップのＷＩＮＤＯＷＳ（登録商標）オペレーティングシステムに関して使用されるものと必ずしも同じ定義であるとは限らないことに留意されたい。それにもかかわらず、当業者は、何がアカウントを構成するのかを理解する。本明細書では、ＳＩＤ１００は、アカウントデータベース内のエントリにマッピングされる、例えばＷＩＮＤＯＷＳ（登録商標）ＣＥオペレーティングシステムといったオペレーティングシステム全体にわたって一意の識別子であるものと仮定される。アカウントデータベース内のエントリは、どのような基本特権及び拡張特権が特定のアカウント（別称ＳＩＤ）に割り当てられているのかを指定する。

【００１４】

[0023]図２は、本発明の一実施形態によるセキュリティトークン構造体２００のレイアウトを示す。セキュリティトークン２００は、識別情報のセット及びそれらの識別情報に割り当てられた特権の集合を定義するのに使用される。通常、セキュリティトークン２００は、プロセス、スレッド、同期のようなランタイムオブジェクト及びメッセージキュー内の個々のメッセージに関連付けられる。セキュリティトークン２００は、この構造体のバージョン２０２、フラグ２０４、オフセット２０６、直接グループ（immediate group）の個数２０７、及びグループＩＤの総数２０８を識別するためのフィールドを含む。セキュリティトークン２００の構造体は、プライマリオーナーＳＩＤ２１０、グループＳＩＤ（複数可）２１２、基本特権（basic privilege）２１４、及び拡張特権（extended privilege）２１６も記憶することができる。

【００１５】

[0024]プライマリＳＩＤ２１０及びグループＳＩＤ（複数可）２１２は、このセキュリティトークン２００を有するオブジェクトに関連付けられる識別情報を定義する。基本特権２１４及び拡張特権２１６は、どのような特権が、このセキュリティトークン２００に関連付けられたオブジェクトに許可されるのかを定義する。基本特権２１４は、セキュリティトークン２００において指定された識別情報に有効な特権のセットである。同様に、拡張特権２１６は、セキュリティトークン２００におけるＳＩＤのリストについて定義されたカスタム特権（custom privilege）である。

【００１６】

[0025]基本セキュリティトークン構造体２００は、このセキュリティトークン２００に関連付けられるプライマリＳＩＤ２１０及びグループＳＩＤ２１２並びにこのセキュリティトークン２００に関連付けられる拡張特権２１６等のセキュリティトークン２００のための拡張（オプション）データへのオフセットポインタ２０６を有する。セキュリティトークン２００における個々の各ＳＩＤ２１０、２１２は、ＳＤが指定されたオブジェクトに、所与のセキュリティトークン２００が所望のアクセスを行うことができるか否かを判断するために、AccessCheck（アクセスチェック）ＡＰＩコールにおいて使用される。また、セキュリティトークン２００における基本特権［［２１２］］２１４及び拡張特権［［２１４］］２１６は、所与のセキュリティトークン２００が特権を必要としていたか否かを判断するために、PrivilegeCheck（特権チェック）ＡＰＩコールにおいて使用される。ＡＰＩ及びセキュリティトークン２００の使用は、本明細書の以下で説明するトークン

ＡＰＩの論述を参照してより詳細に論述される。

【００１７】

[0026]アカウントの文字列表現が作成され、この文字列表現は、セキュリティトークン構造体２００で使用されるDWordにマッピングされる。その場合、セキュリティトークン２００は、システムにおける読み出されたスレッドに関連付けられるオブジェクトである。したがって、あらゆるスレッドは、特定のセキュリティトークン２００で開始する。セキュリティトークン２００は、この特定のスレッドが、特定のアカウントＩＤを有する特定のチャンバーに属することを示すと共に該スレッドが一定のグループのメンバであることも示す識別情報のリストを有する基本トークン構造体２２０を含む。各セキュリティトークン２００は、セットによって表される。したがって、セキュリティトークン構造体は、構造体及び複数のセットを含む。第１のセットはオーナーセットである。それは、一意のＩＤがシステム内の各チャンバー又はシステム内の各アカウントに与えられることを意味する。その場合、ゼロ個又は１つ以上のグループメンバーシップであるグループアカウントＩＤが存在する可能性がある。それらのグループアカウントＩＤは、拡張トークンデータ２２２において与えられる。

【００１８】

[0027]デスクトップ用のセキュリティトークン構造体も同様である。例えば、デスクトップをリセットするために定義された特権又はドライバーをインストールするために定義された特権が存在する可能性がある。したがって、特定のコーラーが特定の特権セットを有する場合、そのコーラーのみが、そのＡＰＩコールを行うことができ、そのコーラーのみがデバイスをリセットするか又はドライバーをインストールすることができる。これらは特有の特権である。基本特権は１つのDWordのみとすることができ、それによって、そのDWordは基本特権［［２１２］］２１４に結び付けられ、そして、拡張トークンデータ２２２の一部である任意の個数の拡張特権［［２１４］］２１６が存在し得る。

【００１９】

[0028]多くの特権２１４、２１６が存在し得るが、特権２１４、２１６は、アクセスチェックがリソースについて動作するように正確に機能する。したがって、特定のＡＰＩが特定の特権を必要とする場合、又は特定のＡＰＩが、特定のコーラーが特定の特権を有することを必要とする場合、コンタクトチェーン内のあらゆるコーラーがその特定の特権を有しなければならない。すべてのコーラーがその特権を有するときにのみ、そのコールは通過することが許可される。あらゆるトークンは、基本トークン構造体２２０と少なくとも同程度の情報を有するべきである。

【００２０】

[0029]拡張トークン構造体２２２はオプションの設定である。したがって、一例として、アカウントは、１つ又は複数のグループのメンバである場合がある。したがって、拡張トークン構造体２２２は可変サイズを有し、すなわちサイズは固定でないのに対して、基本トークン構造体のサイズは固定である。セキュリティトークン２００は、あらゆるプロセスに関連付けられ、それゆえ、プロセス内のあらゆるスレッドは、開始時にそのセキュリティトークン２００を受け取るか又は複製する。セキュリティトークン２００が異なるサーバーにマイグレーションするとき、セキュリティトークン２００は変化する可能性がある。マイグレーションされたセキュリティトークン２００は、そのサーバーのセキュリティトークンとインターセクトされる。実際のデータ構造体は変化しない。

【００２１】

[0030]図３は、本発明の一実施形態によるセキュリティ記述子のレイアウト３００の簡略化された図を示す。セキュリティ記述子３００は、パーミッションをリソースに関連付けるのに使用されるデータ構造体である。特定のリソースについて、セキュリティ記述子３００は、どのようなアカウントが、そのリソースに対してどのようなアクセスを有するのかを定義し、その特定のオブジェクトに関するすべてのルールを定義する。通常、オブジェクトのためのセキュリティ記述子３００は、オーナーＳＩＤ３２０、グループＳＩＤ３３０、及び関連付けられるＡＣＬ３４０を定義する。

【００２２】
[0031]これらのすべてはオプションのエントリであるので、ＳＤ３００は、オーナーＳＩＤ３２０、グループＳＩＤ３３０、及びＡＣＬ３４０が構造体の最後にリストされた可変サイズの構造体として定義される。バージョンフィールド３１０及びフラグフィールド３１２は、どのような値がＳＤ３００に含まれるのかを指定するためにＳＤ構造体３００内に設けられる。フィールド３１４も、ＳＤ構造体３００の合計サイズを定義するために設けられる。
【００２３】
[0032]セキュリティトークン（すなわち、図２の２００）とセキュリティ記述子３００との間の関係によって、アクセス権の制御が決まる。スレッドがリソースにアクセスするとき（例えば、ファイルである写真が例えば/windows/myphotos/に作成される）、［［.］］［［That］］そのコールは最終的にはサーバーに入り、サーバーは、コーラーがこのリソースにアクセスすることができるか否かを判断する。サーバーは、そのコールから、リソースへのアクセス権を定義するセキュリティ記述子３００を構築する。したがって、サーバーが、だれがこのリソースにアクセスすることができるのかを識別するセキュリティ記述子３００を一旦有すると、サーバーは、コーラーのセキュリティトークン（すなわち、図２の２００）を見て、コーラーの識別情報を決定する。次に、サーバーは、セキュリティ記述子において定義された識別情報がＡＣＥのうちの１つにあるか否かを判断しようと試みる。セキュリティ記述子で定義された識別情報がＡＣＥのうちの１つにある場合、サーバーは、その識別情報のアクセス権が何であるのかを判断する。その識別情報のアクセスが拒否される場合、そのコールは拒否される。その識別情報のアクセスが読み出し（read）と書いてある場合、読み出しパーミッションのみがこのコーラーに与えられる。
【００２４】
[0033]したがって、コールとそのコールを受け取った特定のリソースとの間のインターセクションをセキュリティトークンを使用して識別しなければならず、その識別情報がどのようなパーミッションをセキュリティ記述子３００において有するのかを判断しなければならない。オペレーティングシステム内では、多数のアカウントＩＤを設けることができる。プロセス内部で動作を開始するあらゆるスレッドが、そのプロセスのためのアカウントＩＤを有するトークンを継承する。スレッドは、サービスの異なるフィールドをコールするとき、トークンが更新される。
【００２５】
[0034]図４は、本発明の一実施形態によるアクセス制御リスト（ＡＣＬ）４００を示す。ＡＣＬ４００はアクセス制御エントリ（ＡＣＥ）の集合である。ＡＣＥは、ＡＣＥヘッダー４２０、４２２及びＡＣＥに関連付けられるＳＩＤ４３０、４３２を含む。ＡＣＥは、どのようなアクセス権が所与のＳＩＤについて定義されているのかを識別する。図４では、ＡＣＬ４００は、リビジョンフィールド４０２、未使用のフィールド４０４、構造体の合計サイズを定義するフィールド４０６、ＡＣＬの最後におけるＡＣＥの個数を定義するフィールド４０８、及び第２の未使用フィールド４１０を含む。ＡＣＥの個数は事前に定義されていないので、ＡＣＬ４００は可変長構造体である。
【００２６】
[0035]各ＡＣＥは、アカウントＩＤと、そのアカウントＩＤがそのオブジェクトに対してどのようなパーミッションを有するのかとを指定する。例えば、或る特定のＡＣＥは、アカウントＩＤ取得（account ID get）とすることができ、アクセスは、その特定のオブジェクトの読み出し専用である。別の特定のＡＣＥは、アカウントＩＤアドミニストレーターアクセス読み出し／書き込み（account ID administrator access read/write）とすることができ、これはすべての異なるＡＣＥの集合を提供し、それらのＡＣＥのそれぞれは、特定のアカウントについて、どのようなアクセスがこのリソースについて指定されているのかを定義する。
【００２７】
[0036]デスクトップ上では、ＡＣＥは、各エントリについて異なるアクセス権を提供す

る。これらの異なるアクセス権は、Allow（許可）、Deny（拒否）、及びAudit（監査）を含む。ＷＩＮＤＯＷＳ（登録商標）ＣＥオペレーティングシステムでは、権利は、所与のＡＣＥについて付与される。デフォルトの戻り値は、必要とされるアクセスマスクを有するＡＣＥがＡＣＬ内にないときの「拒否」である。

【００２８】

[0037]図５は、本発明の一実施形態によるアクセス制御エントリ（ＡＣＥ）５００のための構造体を示す。各ＡＣＥ５００は、どのようなタイプのアクセスがこのＡＣＥ５１０によっていずれの識別情報（ＳＩＤによって与えられる）５２０に対して許可されているのかを定義する。ＳＩＤ５２０は可変長データアイテムであるので、ＡＣＥ５００も可変長構造体である。ＡＣＥ５００に関連付けられるＳＩＤデータ５２０は、ＡＣＥ構造体５００の最後において開始する。ＡＣＥ５００は、フラグのフィールド５３０、構造体の合計サイズを識別するためのフィールド５３２、及びマスクフィールド５３４も含む。

【００２９】

[0038]図６は、本発明の一実施形態によるセキュリティ記述子（ＳＤ）６００のための構造レイアウトを示す。セキュリティ記述子（ＳＤ）６００はヘッダー６１０を含む。ＳＤヘッダー６１０は、ＳＤ６００のバージョン、フラグ、及びサイズを識別する。ＳＤ６００は、次に、オーナーセキュリティ識別子（ＳＩＤ）６２０及びグループセキュリティ識別子（ＳＩＤ）６３０を含む。残りのデータは、アクセス制御リスト（ＡＣＬ）６４０を形成する。ＡＣＬ６４０は、ＡＣＬヘッダー６４２及び１つ又は複数のアクセス制御エントリ（ＡＣＥ）６５０を含む。ＡＣＬヘッダー６４２は、ＡＣＥ６５０のバージョン、サイズ、及び個数を識別する。各ＡＣＥ６５０は、ＡＣＥヘッダー６５２及び関連付けられるセキュリティ識別子（ＳＩＤ）６５４を含む。

【００３０】

[0039]本発明の一実施形態によるセキュリティインフラストラクチャーは、プロセス又はスレッドが作成されるときに、セキュリティ記述子、セキュリティ識別子、アクセス制御リスト、及びアクセス制御エントリによって提供される。例えば、プロセストークンは、不変であり、プロセスが作成されるときに割り当てられる。デフォルトにより、セキュリティが設けられていない場合、すべてのプロセスは（トークン特権について）等しく扱われ、プロセスには、事前に定義されたシステムトークンが割り当てられる。これが、セキュリティが有効にされていないシステム上でのデフォルトの振る舞いである。このようなシステムでは、トラスト境界は、ユーザーモードとカーネルモードとの間の移行ポイントである。

【００３１】

[0040]プロセストークンは、場合によっては、

・アカウントデータベース内のＩＤにマッピングされるｅｘｅエビデンス（パス、ハッシュ、証明書）、

・アカウントデータベース内の所与のアカウントのための基本特権、

・アカウントデータベース内の所与のアカウントのための拡張特権、

・コーラートークンに基づくグループＩＤのリスト、

のようないくつかのデータポイントの組み合わせとすることができる。

【００３２】

[0041]所与の実行可能ファイル（executable）のエビデンスである最初の情報は、セキュアなローダーコンポーネントによって求められ、この文書の範囲外のものである。この特徴の目的で、エビデンスはアカウントデータベース内のＩＤにマッピングされるものと仮定する。これを前提として、ＯＳは、アカウントデータベース内のアカウント情報からトークンを作成する。このトークンは、プロセスが作成されるときにプロセスオブジェクトに関連付けられ、プロセスの寿命の間中不変のままである。

【００３３】

[0042]スレッドが作成されるとき、スレッドトークンが作成され、スレッドオブジェクトに関連付けられる。デフォルトにより、スレッドトークンは、スレッドのオーナープロ

セスに関連付けられるトークンと同一である。プロセストークンとスレッドトークンとの間の主な相違は、スレッドトークンは、スレッドの寿命を通じて変化する可能性があるのに対して、プロセストークンは、プロセスの寿命を通じて不変のままであるということである。例えば、スレッドトークンは、以下のときに変化する可能性がある。

【００３４】

・所与のトークンを偽装するコール：このコールは、コール側のスレッドのアクティブトークンを、そのコールでImpersonate（偽装）に渡されるセキュリティトークンに変化させる。

【００３５】

・前の偽装をリバート（revert）するコール：このコールは、スレッドトークンを偽装コール前のトークンに更新する。

【００３６】

・現在のプロセスを偽装するコール：このコールは、現在のスレッドトークンを、現在のアクティブプロセスのスレッドトークンとなるように更新する。

【００３７】

・ＡＰＩコールからのスレッド復帰：ＡＰＩコールの復帰のとき、カーネルは、スレッドが復帰している復帰元のＰＳＬコンテキストに関連付けられるすべてのスレッドトークンを自動的に削除する。この場合、スレッドに関連付けられる現在のトークンも、ＡＰＩコール前のトークンに更新される。

【００３８】

[0043]図７は、本発明の一実施形態によるスレッド７００のトークンリストのためのアレンジメントを示す。所与のスレッドのための複数のトークンを管理するために、トークンリスト７００が設けられる。トークンリスト７００は、スレッドに関連付けられるすべてのトークンのリンクリストである。スレッドのための現在のアクティブトークン７１０は、常に、トークンリスト７００の先頭のトークンノードである。換言すれば、トークンリスト７００は、リストに最後に追加されたトークンがスレッドの現在のトークンであるＬＩＦＯキュー（別称スタック）のように振舞う。現在のスレッドに対するすべてのアクセスチェック及び特権チェックは、その特有のトークンのみと、そのスレッドの現在のコンテキストとを使用してハンドリングされる。加えて、スレッドがＡＰＩコールから復帰すると、カーネルは、保護されたサーバーライブラリ（ＰＳＬ）７１２のコンテキストにおいて偽装コールによってこのリストに追加されたあらゆるトークンノードを自動的に削除する。この自動的なリバートによって、ＡＰＩコール時のあらゆる特権リークが防止される。

【００３９】

[0044]例えば、ＷＩＮＤＯＷＳ（登録商標）ＣＥといったオペレーティングシステムでは、スレッドは、対応するＡＰＩコールをハンドリングするサーバープロセス内に移行することによってＡＰＩコールを行うことができる。各スレッドは、関連付けられるトークンを、そのトークンについて定義された識別情報／特権のセットと共に有するので、ＡＰＩサーバーは、ＡＰＩコールが現在のスレッド上で許可されるか否かを判定するのに、スレッドの現在の特権を考慮する必要がある。ＡＰＩサーバーがＡＰＩコールを完了するのに使用することができるトークンには２つの可能なトークンがある。

【００４０】

・コーラーのトークン：この場合、ＡＰＩサーバーは、CeAccessCheck（GetCurrentToken()，…）をコールする。これは、ＡＰＩコールが高い特権を有するチャンバーから低い特権を有するチャンバー内へ入る場合に、潜在的なセキュリティリスクになる。実際には、高い特権という概念も低い特権という概念もない。しかしながら、この論述の目的で、高い特権チャンバーは、低い特権チャンバーのすべての特権及びいくつかの追加の特権を有するものと仮定する。

【００４１】

・現在のプロセストークン：この場合、ＡＰＩサーバーは、現在のスレッドのトークン

が現在のプロセス（ＡＰＩサーバーのトークン）となるように更新されるCeAccessCheck（ImpersonateCurrentProcess(), …）をコールする。これも、ＡＰＩコールが低い特権を有するチャンバーから高い特権を有するチャンバー内へ入る場合に、潜在的なセキュリティリスクになる。

【００４２】

[0045]あらゆるスレッドが、最初にオーナープロセストークン７２０で開始する。したがって、オーナープロセストークン７２０は、スレッド７００のためのトークンレイアウトを示す図７の下部にリストされている。スレッドは、異なるコーラーから又はメッセージキューから非同期にトークンを受け取ることができる。さらに、スレッドがそのトークンに基づいてリソースにアクセスしたいとき。スレッドは、プロセストークンを使用したくないが、代わりに、自身の識別情報を変更したい。したがって、スレッドは、ＡＰＩのうちの１つをコールしてそのＡＰＩを偽装することができる。このＡＰＩは、そのトークンをトークンリストの最上部にプッシュする。したがって、スレッドがトークンをコールし、そのトークンを偽装するときはいつでも、そのトークンはリストの最上部になる。図７では、偽装されたトークン７３０が、オーナープロセストークン７２０の上に示されている。図７には、Ｎ個の偽装されたトークン７３０～７５０のチェーンが示されている。

【００４３】

[0046]スレッドがアクセスするどのリソースも、最上部のトークン７５０、すなわち現在のトークン７１０内のトークンに対してチェックされる。偽装が起動されるときはいつでも、偽装トークン７５０が、常にオーナープロセストークン７２０とインターセクトされる。したがって、スレッドがリソースにアクセスするときはいつでも、オーナープロセストークン７２０は、そのリソースにおいてアクセス権を提示しなければならず、偽装されただけのトークン７５０も、そのリソースにアクセス可能でなければならない。このように、スレッドは、異なるサーバーへ移行しているときに異なるサーバーにＡＰＩコールを行うことができる。加えて、コンタクトチェーン上にいくつかのサーバーが存在する場合、すべてのサーバーは、そのリソースにアクセス可能でなければならない。

【００４４】

[0047]３つのすべてのチェックが検証された場合、スレッドへのアクセスが許可される。重要なことに、このプロセスは、個々の特権を提供することとは異なる。例えば、動作が現在のトークン７５０にのみ基づいている場合でかつ現在のトークン７５０が、より高い特権を偶然有する場合には、それによって、特権が昇格される。逆に、動作がコーラーのトークン７２０にのみ基づいている場合には、その動作は、高い特権サーバーから低い特権サーバーへマイグレーションされたものである。したがって、どちらの場合にも、個々の特権のみが使用されるとき、特権の可能な昇格に起因してセキュリティリスクがもたらされる。

【００４５】

[0048]一旦ＡＰＩコールが完了し、ＡＰＩコールが復帰されると、トークンは、あらゆる特権反復（iteration of privilege）のリークを防止するためにサーバー内で行われる任意の偽装によってリバートされる。したがって、ＡＰＩコールが復帰するときは、たとえサーバーが、自身が偽装したトークンをリバートするのを忘れても、サーバーは自動的にリバートされる。このように、クライアントも、自身がＡＰＩコール前に何であろうとも、ＡＰＩコールがそのトークンを返すときは、ＡＰＩコール前のものに常にリバートして戻る。

【００４６】

[0049]ご覧のように、コーラーのスレッドトークンのみ又は現在のプロセススレッドトークンのみを使用する場合にはセキュリティ問題が存在する。この問題を解決するために、本発明者らは、所与のスレッドによるリソースに対するアクセスをチェックするときに以下のことを考慮することを提案している。

【００４７】

・スレッドの現在のトークンに関連付けられるすべての識別情報がリソースにアクセス

可能であるべきである。

【００４８】

・現在のスレッドコールスタック内のすべてのチャンバーに関連付けられるすべての識別情報（偽装境界まで）がリソースにアクセス可能であるべきである。

【００４９】

・そのスレッドに関する保存されたコンテキストに関連付けられるすべての識別情報がリソースにアクセス可能であるべきである。このチェックは、主として、アクセスチェックが、コール側スレッド以外の異なるスレッド内のコーラーのために実行されるときに使用される。

【００５０】

[0050]この変更は、アクセス／特権が所与のスレッド及びオブジェクトについてどのようにチェックされるのかに影響を与える。コードが任意のオブジェクトにアクセスするには、スレッドの現在のトークン７５０及びＡＰＩサーバーのトークン、すなわちオーナープロセス／スレッドトークン７２０の双方が、そのオブジェクトにアクセス可能であるべきである。コードが任意の特権を取得するには、スレッドの現在のトークン７５０及びＡＰＩサーバーのトークン７２０（この例では）がその特権セットを有するべきである。その結果、スレッドの現在のトークン７５０は、現在のスレッドのコールスタックチェーン７００と常に組み合わせられるので、コードは、デフォルトにより、常に最小アクセス／特権セットで動作する。これによって、アクセス／特権チェックを現在のスレッドに対して実行するときに、コーラーのトークン７２０のみ又は現在のプロセストークン７５０のみのいずれかを使用することから生じるセキュリティ問題をハンドリングするセキュアなプロセスが提供される。トークンＡＰＩのいくつかの論述が、本明細書において表１を参照して以下で提供される。

【００５１】

【表１】

名称	引数	戻り値
CeCreateToken(pToken, flags)	トークン構造体へのポインタ、フラグ	トークンへのハンドル
CeImpersonateToken(hToken)	トークンのハンドル	偽装が成功した場合にはＴＲＵＥ、そうでない場合にはＦＡＬＳＥ
CeImpersonateCurrentProcess(void)	なし	現在のスレッドのトークンが更新された場合にはＴＲＵＥ、そうでない場合にはＦＡＬＳＥ
CeRevertToSelf(void)	なし	なし
CeGetProcessAccount(hProcess)	プロセスのハンドル	所与のプロセスに関連付けられるオーナーアカウント
CeGetThreadAccount(hThread)	スレッドのハンドル	所与のスレッドに関連付けられるオーナーアカウント
CeGetOwnerAccount(hToken)	トークンのハンドル	所与のトークンに関連付けられるオーナーアカウント
CeGetGroupAccount(hToken, idx)	トークンのハンドル、及びグループアカウントのインデックス	所与のトークンに関連付けられるグループアカウント
CeAccessCheck(pSD, hToken, AccessRequired)	ＳＤへのポインタ、トークンへのハンドル、及び必要とされるアクセスマスク	所与のトークンが所与のＳＤに所望のアクセスを行うことができる場合にはＴＲＵＥ、そうでない場合にはＦＡＬＳＥ このＡＰＩは、所与のトークンにおけるの各アカウントについて及び所与のＳＤ内にリストされた各ＡＣＥについて所与のアクセスをチェックする
CePrivilegeCheck(hToken, pPrivileges, cPrivileges)	トークンへのハンドル、チェックする特権のアレイ、及び特権の個数	所与のトークンが特権を要求していた場合にはＴＲＵＥ、そうでない場合にはＦＡＬＳＥ このＡＰＩは、所与のトークンの所与の特権アクセスをチェックする。ＡＰＩが成功するには、（pPrivilegesアレイで指定された）すべての所与の特権が、トークンの基本特権リスト又は拡張特権リスト内に存在しなければならない。
GetCurrentToken(void)	なし	スレッドの現在トークンへの疑似ハンドル

【００５２】

[0051]以下の説明は、コア偽装ＡＰＩ及びアクセスチェックＡＰＩについてのものである。表１にリストされた最初のトークンはCeCreateToken(pToken, flags)であり、(pToken, flags)フィールドはトークン構造体へのポインタ及びフラグを提供する。トークンへのハンドルが返される。

【００５３】

[0052]CeImpersonateTokenは、任意のアプリケーションが所与のトークンを偽装するのに使用することができる。これは、コーラーのために非同期コールを実行するのにも使用される。このＡＰＩのための実施は、このトークンを現在のスレッドのトークンリストのためのスタックの最上部にプッシュし、このトークンをそのスレッドの現在のトークンにすることである。このＡＰＩコールは、現在のプロセストークンを、このＡＰＩコールへの指定されたトークンと常にマージし、マージされたトークンは現在のスレッドのトークンとして扱われる。

【００５４】
[0053]加えて、（現在のプロセストークンによって与えられた）現在のプロセス識別情報は、トークン識別情報の過去のもの（the passed）に暗黙的に追加される。これを行う主な理由は、コーラーがこのＡＰＩコールで任意の特権を取得することを防止するためである。換言すれば、このＡＰＩは、降格することに制限されるか、又は多くとも現在のプロセスと同じ特権を有する。これによって、デフォルトにより最小特権レベルでのコード実行が可能になる。その使用法は次の通りである。すなわち、ＡＰＩコール時に、現在のスレッドのトークンはスタッシュされ、そのＡＰＩコールをコール時のコーラーのコンテキストで実行するためにＡＰＩサーバーいおいて非同期スレッドによって後に使用される。

【００５５】
[0054]CeImpersonateCurrentProcess(void)トークンは、現在のスレッドのトークンが更新される場合にはＴＲＵＥを返し、そうでない場合にはＦＡＬＳＥを返す。CeRevertToSelfトークンは、現在のスレッドのトークンリストのトークンを「ポップ」するのに使用される。通常、これは、CeImpersonateTokenＡＰＩコール又はCeImpersonateCurrentProcessＡＰＩコールを介して行われたあらゆるトークン偽装をＡＰＩサーバーがアンドゥーするのに使用される。CeImpersonateCurrentProcessトークンは、現在のスレッドのためのトークンリストを現在のプロセストークンにトランケートするのに使用される。これは、現在のプロセストークンを現在のスレッドのトークンリストの先頭に効果的に「プッシュ」する。

【００５６】
[0055]CeAccessCheckトークンは、３つの引数を取り込む。第１の引数は、アクセスを要求しているオブジェクトのためのパーミッションセットを表すＳＤである。第２の引数は、オブジェクトへのアクセスを要求する識別情報であるトークンである。第３の引数は、オブジェクトへの所望のアクセスである。CeAccessCheckトークンは、所与のトークンにおける各アカウントについて及び所与のＳＤにリストされた各ＡＣＥについて所与のアクセスをチェックするのに使用される。このＡＰＩは、所与のトークンにおける各アカウントについて及び所与のＳＤにリストされた各ＡＣＥについて所与のアクセスをチェックする。

【００５７】
[0056]CePrivilegeCheckトークンは、所与のトークンにおける所与の特権アクセスをチェックする。このＡＰＩは、所与のトークンにおける所与の特権アクセスをチェックする。ＡＰＩが成功するには、（pPrivilegesアレイで指定された）すべての所与の特権が、トークンの基本特権リスト又は拡張特権リスト内に存在しなければならない。

【００５８】
[0057]CeGetProcessAccount(hProcess)トークンは、所与のプロセスに関連付けられるオーナーアカウントを返す。CeGetThreadAccount(hThread)トークンは、所与のスレッドに関連付けられるオーナーアカウントを返す。CeGetOwnerAccount(hToken)トークンは、所与のトークンに関連付けられるオーナーアカウントを返す。CeGetGroupAccount(hToken, idx)トークンは、トークンのハンドル及びグループアカウントのインデックスを使用して、所与のトークンに関連付けられるグループアカウントを返す。GetCurrentToken (void)トークンは、疑似ハンドルをスレッドの現在のトークンに提供する。

【００５９】
[0058]図８は、本発明の一実施形態によるＰＳＬコール８００のためのスレッドのためのコールスタックリストを示す。動作中、カーネルは、スレッドごとにそのスレッドと共にコールスタック構造体のリストを保持する。ＰＳＬコール及びＰＬＳコール復帰ごとに、スレッドのためのコールスタックリスト８００が更新される。図８では、４つのコールスタック８１０、８２０、８３０、８４０が示されている。最後のコールスタック８４０は、最上部にあり、現在のコールスタックである。ＰＬＳコールエントリ時に、新しいコールスタックが追加される。ＡＰＩコール復帰時には、最上部のコールスタック構造体が

削除される。コールスタック構造体８００は、スレッドがマイグレーションした１つのＰＳＬサーバーに対応する。

【００６０】

[0059]図９は、本発明の一実施形態によるスレッドのための、偽装リストとコールスタックリストとの間のリンク９００を示すブロック図である。コールスタック構造体と同様に、スレッドのためのトークンは、トークンリストの最上部に、現在の偽装を有するリスト内に保持される。図９には、４つのコールスタック９１０、９２０、９３０、９４０が示されている。最後のコールスタック９４０は、最上部にあり、現在のコールスタックである。ＰＳＬコールにわたって特権リークがないことを確実にするために、カーネルは、現在のスレッドのための偽装リストと現在のスレッドのためのコールスタックリストとの間のリンクを保持する。各偽装ノードは、その偽装コールが発生したコールスタック構造体に関連付けられる。

【００６１】

[0060]したがって、図９では、アプリケーションに属するスレッドがＰＳＬサーバーＳ１　９１０にマイグレーションしている。ＰＳＬサーバーＳ１は、偽装ＡＰＩをコールし、その結果、偽装ノード９１２が偽装リストに追加され、関連付けられるコールスタックＳ１が、偽装ノードＴ１　９１２においてマーキングされる。次に、ＰＳＬサーバーＳ１がＰＳＬサーバーＳ２　９２０にマイグレーションする。Ｓ２は、偽装コールを行わない。次に、スレッドが、同じコールでＳ３　９３０にマイグレーションし、Ｓ３サーバーが偽装トークンをコールするとき。新しい偽装ノード９３２がトークンリストに追加され、関連付けられるコールスタック構造体Ｓ３　９３０がＴ２　９３２に記される。

【００６２】

[0061]リソースチェックは、スレッドの寿命の異なる段階で実行される。例えば、Ｓ１　９１０からのアクセスチェックを考える。この場合、リソースは、スレッドがＳ１プロセス９１０にマイグレーションしたときにアクセスについてチェックを受けることになる。以下のチェックが実行される。

【００６３】

・トークンＴ１におけるすべての識別情報がリソースにアクセス可能であるべきである。

【００６４】

・プロセスＳ１に関連付けられるトークンにおけるすべての識別情報がリソースにアクセス可能であるべきである。

【００６５】

[0062]次に、Ｓ２　９２０からのアクセスチェックを考える。この場合、リソースは、スレッドがＳ１　９１０プロセスを介してＳ２　９２０プロセスにマイグレーションしたときにアクセスについてチェックを受けることになる。この場合、以下のチェックが実行される。

【００６６】

・トークンＴ１におけるすべての識別情報は、リソースにアクセス可能であるべきである。

【００６７】

・プロセスＳ２に関連付けられるトークンにおけるすべての識別情報がリソースにアクセスできるべきである。

【００６８】

・プロセスＳ１に関連付けられるトークンにおけるすべての識別情報がリソースにアクセスできるべきである。

【００６９】

[0063]Ｓ３　９３０からのアクセスチェックを実行する際、スレッドは、Ｓ１　９１０を介しＳ２　９２０を介してプロセスＳ３　９３０にマイグレーションする。アクセスチェックは、Ｓ３　９３０からコールされたものである。以下のチェックが実行される。

【００７０】
・トークンＴ２におけるすべての識別情報がリソースにアクセス可能であるべきである。
【００７１】
・プロセスＳ３に関連付けられるトークンにおけるすべての識別情報がリソースにアクセス可能であるべきである。
【００７２】
[0064]この場合、たとえスレッドのフルコンテキストがプロセスＳ２　９２０及びＳ１　９１０をそのコールチェーンに有していても、Ｓ３　９３０はトークンを偽装し、偽装はコールスタックチェーンを効果的に切断するので、アクセスチェックは、これらのプロセスに対して行われないことに留意されたい。
【００７３】
[0065]（コールスタックの最上部９４０としてリストされた）現在のプロセスからのアクセスチェックを実行する際、スレッドコールスタックチェーン９００におけるＳ３プロセス９３０と現在のプロセス９４０との間にもはやトークン偽装はない。以下のチェックが実行される。
【００７４】
・トークンＴ２におけるすべての識別情報がリソースにアクセス可能であるべきである。
【００７５】
・コールスタックチェーンにおける現在のプロセス対プロセスＳ３に関連付けられるトークンにおけるすべての識別情報がリソースにアクセス可能であるべきである。
【００７６】
[0066]したがって、アクセスチェックの基本ルールは、現在のスレッドのためのトークンリストの先頭のトークンである現在のトークンがリソースにアクセス可能であるべきか否か、及び偽装をコールした現在のプロセスから最後のプロセスまでのすべてのプロセス（現在のプロセス及び最後のプロセスを含む）がリソースにアクセス可能であるべきか否かを含む。
【００７７】
[0067]CeGetAcccessMaskＡＰＩは、所与のＳＤのトークンに割り当てられた最大アクセスを返すために提供することができる。このようなＡＰＩは、３つの引数、すなわち
・アクセスをチェックしたい識別情報のリストを有するトークン
・所与のリソースのための異なるＩＤに関連付けられるＡＣＥをリストするセキュリティ記述子
・所与のＳＤ及びトークンの組み合わせに関する最大アクセスセットを返す［出力］パラメーター（[out] parameter）
を使用して実施することができる。
【００７８】
[0068]このＡＰＩの実施は簡単である。すなわち、トークンにおける各ＩＤについて、ＳＤ内にAllow ACEが存在するか否かを判断するチェックが実行される。存在する場合、そのＡＣＥに関連付けられるアクセスマスクが、返されるアクセスマスク値に追加される。このステップは、現在のスレッドコンテキストの各トークンについて（最後の偽装まで）繰り返され、すべてのトークンにおいて設定されたそのアクセスマスクのみを返す。CeAccessCheckとは異なり、このＡＰＩコールは、ＳＤ内のすべてのＡＣＥをスキャンして、一致するＩＤを有するすべてのアクセスマスクの和集合をトークンから得る必要があることに留意されたい。
【００７９】
[0069]加えて、CeDuplicateTokenＡＰＩは、以下の引数を取り込むことによって提供することができる。
【００８０】

・コーラーがコピーしようと試みているソーストークンへのハンドル
・コール側プロセス内の新しくコピーされたトークンオブジェクトのコピーのトークンオブジェクトへのハンドルへの［出力］ポインタ

[0070]図１０は、本発明の一実施形態によるメッセージキューシステム１０００を示す。メッセージキュー１０１０は、本発明の一実施形態によるクライアント１０２０のプロセスとサーバー１０３０のプロセスとの間の非同期メッセージングを提供する。このメッセージキュー１０１０によって、クライアント１０２０は、メッセージキュー１０１０の書き込み側をオープンすることが可能になり、サーバー１０３０は、メッセージキュー１０１０の読み出し側をオープンすることが可能になる。クライアント１０２０及びサーバー１０３０は、メッセージ及び他のデータをメッセージキュー１０１０にポストすることができる。複数のメッセージキュー１０１０を設けることができ、このようなメッセージキュー１０１０は非同期に動作する。したがって、クライアント１０２０がメッセージを書き込み、その後去ると、クライアント１０２０は、肯定応答も、どのタイプの信号も待つことはない。サーバー１０３０は、その後、メッセージを取り出し、リトリーブしたメッセージに基づいて或る動作を実行することができる。サーバー１０３０がメッセージを取り出すとき、サーバー１０３０は、そのメッセージを書き込んだ時点においてそのメッセージを実際に保持するスレッドのセキュリティクレデンシャルを有しなければならない。コンタクトは、サーバー１０３０がそのメッセージを読み出す時までに変化する可能性があり、したがって、サーバー１０３０がメッセージを読み出すとき、サーバーが実際にコンタクトスナップショットにアクセスすることができ、そのスナップショットを偽装し、その後、メッセージを作成して結果を返信することができるように、コンタクトスナップショットのコピーが作成されてメッセージと共に書き込まれていなければならない。

【００８１】

[0071]メッセージ及びそのメッセージを送信した者の識別情報を、サーバー１０３０がそれらのために動作することができるように、サーバー１０３０に非同期に渡すプロセスに加えて、メッセージ及びそのメッセージのポスターの識別情報、並びにメッセージがポストされた時点で存在するメッセージのポスターのセキュリティコンタクト全体も、プロセスの一部である。このように、よりセキュアな判定をその後になって行うことができる。しかしながら、当業者は、本発明が、本明細書で説明したようなメッセージキューを使用することに限定されるよう意図されていないことを認識する。それどころか、スレッドのためのコンタクトチェーンをサーバーに提供するための他の技法も可能である。

【００８２】

[0072]プロセスが開始するとき、そのプロセスには或る識別情報が割り当てられ、そのプロセスにおけるスレッドのそれぞれにその識別情報が与えられる。スレッドがシステム上の他のプロセスにマイグレーションするとき、これらのスレッドは、該スレッドに関するセキュリティ判定が開始されるときに、それらの識別情報のすべてをインターセクトしてスレッドのアクセス特権を求めることができるように、それらの他のプロセスの識別情報を累積する。さらに、プロセスは、どの特定のサーバーにも特有のものでないように意図されている。それどころか、どのサーバーもこのモデルを使用することができる。例えばＷＩＮＤＯＷＳ（登録商標）オペレーティングシステム上では、データトークンが入力される場合、特定の特権、すなわち偽装特権が存在しなければならない。偽装の結果、最悪の場合に権利が降格されることから、偽装は許可される。

【００８３】

[0073]例えばデスクトップ上での複数のコーラーを伴うシナリオでは、アプリケーションが、オペレーティングシステムによってエクスポートされるファイル作成（create file）又は他の任意のＡＰＩをコールするとき、スレッドは、カーネルモードにおいてユーザーの境界で停止し、カーネル内部の別のスレッドが、そのＡＰＩのための動作をスピンオフして実行し、その後、そのコールをスレッドに返す。したがって、スレッドは、デスクトップ上において、実際には或るプロセスから別のプロセスに移行することになる。それに対して、ＷＩＮＤＯＷＳ（登録商標）CEオペレーティングシステム上では、アプリケ

ーションがＡＰＩコールを行うとき、スレッドは、実際にはサーバープロセスに移行する。

【００８４】

[0074]通常、サーバープロセスは、クライアントよりも高い特権である。したがって、アプリケーションがレジスタオープンキーをコールして、レジストリをオープンし、その特定のレジストリが、低い特権アプリケーションが読み出すことができない保護されたキーである場合、スレッド上のそのコール及びそのスレッドは、レジストリＡＰＩを実際に実施するサーバープロセスにマイグレーションする。そのサーバープロセスがアクセスキーを携行している場合、これは、コーラーのセキュリティコンタクトが考慮されていないので、その同じ種類の振る舞いとなる。一方、コーラーのセキュリティコンタクトが現在のセキュリティコンタクトと比較される場合、その保護されたリソースへのアクセスは拒否される。

【００８５】

[0075]Ｗｉｎｄｏｗｓ　Ｍｏｂｉｌｅ（登録商標）オペレーティングシステムのスレッドは実際にマイグレーションし、セキュリティコンタクト情報はスレッドと共に保持されなければならないので、デスクトップＷＩＮＤＯＷＳ（登録商標）オペレーティングシステムとＷｉｎｄｏｗｓ　Ｍｏｂｉｌｅ（登録商標）オペレーティングシステムとは異なる。その時になって初めて、コンタクト全体に基づいてリソースへのアクセスがいつ要求されたのかの判定を行うことができる。デスクトップの場合、ユーザーモードからカーネルモードへ移行する時点で、現在のスレッドコンタクトをコピーしなければならず、その後、カーネルスレッド上で、コーラーが移行され、動作は、あたかもそのコーラーのために完了されたかのように完了される。動作が一旦完了されると、元のセキュリティコンタクトへ戻る移行が実行され、その結果がコーラーに返される。

【００８６】

[0076]特定のスレッドのためのセキュリティコンタクトは、シリアル化及びデシリアル化される。例えばＷｉｎｄｏｗｓ　Ｍｏｂｉｌｅ（登録商標）オペレーティングシステムでは、データベースレコードを更新することができる者及び編集することができる者を制限するように、データベースレコードが保護される。セキュリティを提供するために、レコードのセキュリティコンタクトがレコードと共に保存される。したがって、例えば、レコードを更新するコールが行われたとき、コーラーが、要求されたデータベースオペレーションを実行するのに十分なアクセスを有するか否かを調べるために、そのレコードのセキュリティコンタクト及びコーラーも伝達することができる。このプロセスを成し遂げるために、データベースは永続的であることから、セキュリティコンタクトは、オフラインデータベースにおいて処理可能でなければならない。したがって、Ｗｉｎｄｏｗｓ　Ｍｏｂｉｌｅ（登録商標）オペレーティングシステムを実行しているデバイスが設定され、そのデバイスが復帰するときは、データベースレコードが同じコンタクトをまだ保存しているか否かの判断が行われる。コーラーが識別されたとき、そのコンタクトチェーン内のすべてのアカウントが求められ、その後、ファイルとして保存される。新しいコールが行われたとき、コーラーがコンタクトチェーン内のアカウントのサブセットであることを保証するために、ファイルが再び構築され、保存されたファイルと比較される。

【００８７】

[0077]アプリケーションがサーバーをコールした場合、サーバーは、或るスレッドにおいて動作を実行する。異なるスレッドでは、すべてのリソースがコーラーのためにアクセスされるように、コーラーを偽装することができる。クライアントからのセキュリティコンタクトのスナップショットは、基本的には、送信者の名称、現在のコンタクトチェーン内のすべてのコーラー、どのようなアカウント又は異なるアカウントがコンタクトチェーン内にあるのか、及び異なるサーバーへのマイグレーションが行われたか否かを読み出すことを含む。例えば、アプリケーション４ＲＸＥは、或るサーバーをコールしており、その時、その同じスレッドが別のサーバーをコールする可能性がある。したがって、複数のサーバーがコンタクトチェーン上に存在する可能性がある。しかしながら、それらのどの

１つも、適切にアクセス可能でなければならない。したがって、コンタクトスナップショットがとられたとき、現在コンタクトにあるすべてのアカウントがキャプチャされなければならない。

【００８８】

[0078]特に、スレッドがシステムを通じてマイグレーションするとき、スレッドは異なるサーバーを通過して異なる権利を有するので、そのスレッドのセキュリティコンタクトは変化する。それらの権利のすべてのインターセクションを求めなければならない。セキュリティ判定を行うには、一時点のスナップショットをとることができる必要がある。換言すれば、セキュリティチェックが行われるとき、チェックを非同期又は後に実行することができるように、検証されるすべてのＩＤが保存される必要がある。

【００８９】

[0079]上述したシステム及びコンポーネントは、ネットワーク接続環境、分散環境、又は他のコンピューター実施される環境の一部として実施することができる。これらのシステム及びコンポーネントは、有線通信ネットワーク、無線通信ネットワーク、及び／又はそれらの通信ネットワークの組み合わせを介して通信することができる。デスクトップコンピューター、ラップトップ、ハンドヘルド、又は他のスマートデバイスを含む複数のクライアントコンピューティングデバイスがこのシステムとインターラクトすることができかつ／又はこのクライアントコンピューティングデバイスをこのシステムの一部として含めることができる。代替的な実施形態では、所望の実施態様に従ってさまざまなコンポーネントを組み合わせかつ／又は構成することができる。他の実施形態及び構成も利用可能である。

【００９０】

[0080]次に図１１を参照して、以下の論述は、本発明の実施形態を実施することができる適したコンピューティング環境の簡潔で一般的な説明を提供するように意図されている。本発明は、パーソナルコンピューターのオペレーティングシステム上で動作するプログラムモジュールと共に実行されるプログラムモジュールの一般的なコンテキストで説明されるが、当業者は、本発明が、他のタイプのコンピューターシステム及びプログラムモジュールと組み合わせて実施することもできることを認識するであろう。

【００９１】

[0081]一般に、プログラムモジュールには、特定のタスクを実行するか又は特定の抽象データタイプを実施するルーチン、プログラム、コンポーネント、データ構造体、及び他のタイプの構造体が含まれる。その上、当業者は、ハンドヘルドデバイス、マルチプロセッサシステム、マイクロプロセッサベースの民生用電子機器又はプログラマブル民生用電子機器、ミニコンピューター、メインフレームコンピューター等を含む他のコンピューターシステム構成で本発明を実施することができることを認識するであろう。本発明は、タスクが、通信ネットワークを通じてリンクされたリモート処理デバイスによって実行される分散コンピューティング環境でも実施することができる。分散コンピューティング環境では、プログラムモジュールをローカルメモリストレージデバイス及びリモートメモリストレージデバイスの双方に配置することができる。

【００９２】

[0082]次に図１１を参照して、本発明の実施形態の実例となる動作環境を説明する。図１１に示すように、コンピューター１１００には、汎用のデスクトップコンピューター、ラップトップコンピューター、ハンドヘルドコンピューター、又は１つ又は複数のアプリケーションプログラミングを実行することができる他のタイプのコンピューターが含まれる。コンピューター１１００は、少なくとも１つの中央処理装置１１０８（「ＣＰＵ」）、ランダムアクセスメモリ１１１８（「ＲＡＭ」）及び読み出し専用メモリ（「ＲＯＭ」）１１２０を含むシステムメモリ１１１２、並びにメモリをＣＰＵ１１０８に結合するシステムバス１１１０を含む。スタートアップ中等にコンピューター内のエレメント間で情報を転送するのに役立つ基本ルーチンを含む基本入出力システムが、ＲＯＭ１１２０内に記憶される。コンピューター１１００は、オペレーティングシステム１１３２、アプリケ

ーションプログラム、及び他のプログラムモジュールを記憶するためのマスストレージデバイス１１１４をさらに含む。

【００９３】

[0083]マスストレージデバイス１１１４は、バス１１１０に接続されたマスストレージコントローラ（図示せず）を通じてＣＰＵ１１０８に接続される。マスストレージデバイス１１１４及びそれに関連付けられるコンピューター可読媒体は、コンピューター１１００のための不揮発性ストレージを提供する。本明細書に含まれるコンピューター可読媒体の説明は、ハードディスク又はＣＤ－ＲＯＭドライブ等のマスストレージデバイスを指すが、コンピューター可読媒体は、コンピューター１１００がアクセス又は利用することができる任意の利用可能な媒体とすることができることが当業者によって認識されるべきである。

【００９４】

[0084]限定ではなく例として、コンピューター可読媒体には、コンピューターストレージ媒体及び通信媒体を含めることができる。コンピューターストレージ媒体には、コンピューター可読命令、データ構造体、プログラムモジュール、又は他のデータ等の情報の記憶のための任意の方法又は技術で実施される揮発性及び不揮発性の着脱可能及び着脱不能な媒体が含まれる。コンピューターストレージ媒体には、ＲＡＭ、ＲＯＭ、ＥＰＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリ、若しくは他のソリッドステートメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（「ＤＶＤ」）、若しくは他の光学ストレージ、磁気カセット、磁気テープ、磁気ディスクストレージ、若しくは他の磁気ストレージデバイス、又は所望の情報を記憶するのに使用することができかつコンピューター１１００がアクセスすることができる他の任意の媒体が含まれるが、これらに限定されるものではない。

【００９５】

[0085]本発明のさまざまな実施形態によれば、コンピューター２は、例えばローカルネットワーク、インターネット等のネットワーク１１０４を通じたリモートコンピューターへの論理接続を使用してネットワーク接続環境で動作することができる。コンピューター１１００は、バス１１１０に接続されたネットワークインターフェースユニット１１１６を通じてネットワーク１１０４に接続することができる。ネットワークインターフェースユニット１１１６は、他のタイプのネットワーク及びリモートコンピューティングシステムに接続するのに利用することもできることが認識されるべきである。コンピューター１１００は、キーボード、マウス等（図示せず）を含む複数の他のデバイスからの入力を受け取って処理するための入出力コントローラー１１２２も含むことができる。同様に、入出力コントローラー１１２２は、表示スクリーン、プリンタ、又は他のタイプの出力デバイスに出力を提供することもできる。

【００９６】

[0086]上記で簡略的に述べたように、ワシントン州レドモンドのマイクロソフト社（Microsoft Corporation）が提供しているＷＩＮＤＯＷＳ（登録商標）オペレーティングシステム等、ネットワーク接続されたパーソナルコンピューターのオペレーションを制御するのに適したオペレーティングシステム１１３２を含めて、複数のプログラムモジュール及びデータファイルをコンピューター１１００のマスストレージデバイス１１１４及びＲＡＭ１１１８に記憶することができる。マスストレージデバイス１１１４及びＲＡＭ１１１８は、１つ又は複数のプログラムモジュールも記憶することができる。特に、マスストレージデバイス１１１４及びＲＡＭ１１１８は、クライアントアプリケーションプログラム１１４０及び他のソフトウェアアプリケーション１１４２を記憶することができる。図１１に示すようなコンピューター１１００は、図１～図１０で説明したような本発明の実施形態によるセキュリティインフラストラクチャーを提供する命令を実行するように構成することができる。

【００９７】

[0087]本発明のさまざまな実施形態は、（１）コンピューター実施されるアクト若しくはコンピューティングシステム上で動作するプログラムモジュールのシーケンスとしてか

つ／又は（２）コンピューティングシステム内の相互接続されたマシン論理回路若しくは回路モジュールとして実施できることが認識されるべきである。この実施は、本発明を実施するコンピューティングシステムの性能要件に応じた選択の問題である。したがって、関係したアルゴリズムを含む論理オペレーションは、オペレーション、構造デバイス、アクト、又はモジュールとさまざまに呼ぶことができる。これらのオペレーション、構造デバイス、アクト、及びモジュールは、本出願で述べられた特許請求の範囲内に挙げられた本発明の趣旨及び範囲から逸脱することなく、ソフトウェア、ファームウェア、専用デジタルロジック、及びそれらの任意の組み合わせで実施できることが当業者によって認識される。

【００９８】

[0088]本発明を、さまざまな例示の実施形態に関して説明してきたが、当業者は、次の特許請求の範囲の範囲内においてそれらの実施形態に多くの変更を行えることを理解する。したがって、本発明の範囲が上記説明によっていかなる形においても限定されることは意図されておらず、それよりむしろ、専ら次の特許請求の範囲を参照することによって判断されることが意図されている。

【図１】

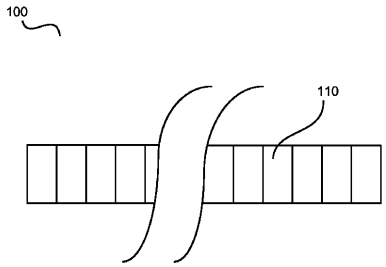
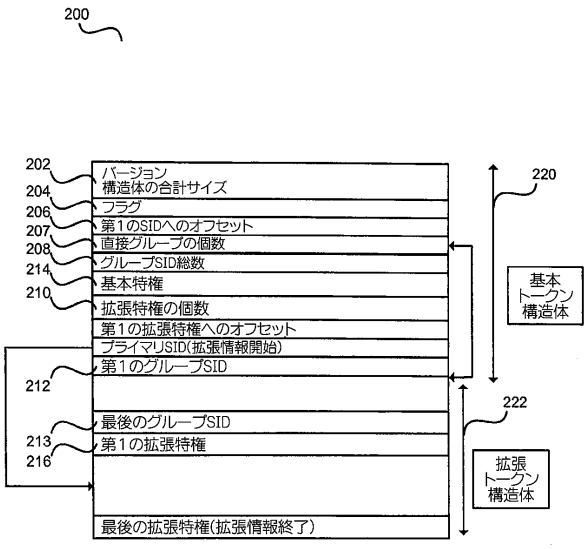
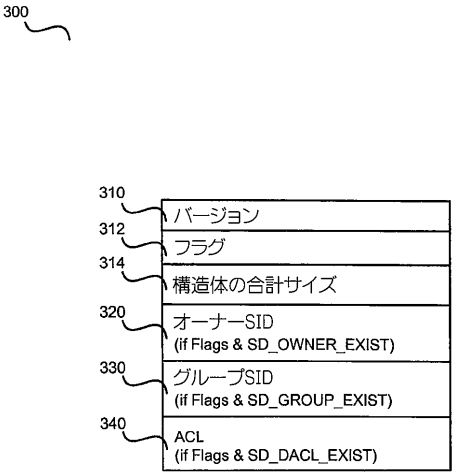


FIG. 1

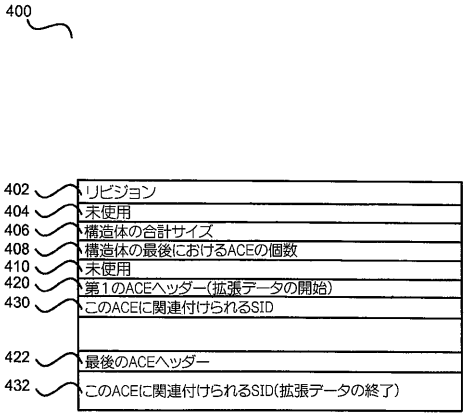
【図２】



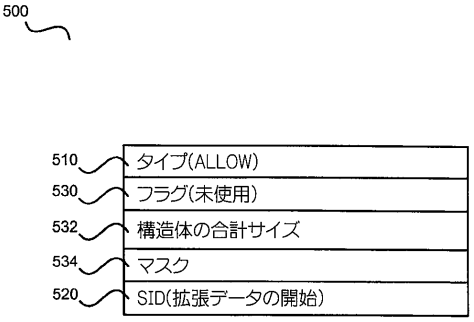
【図３】



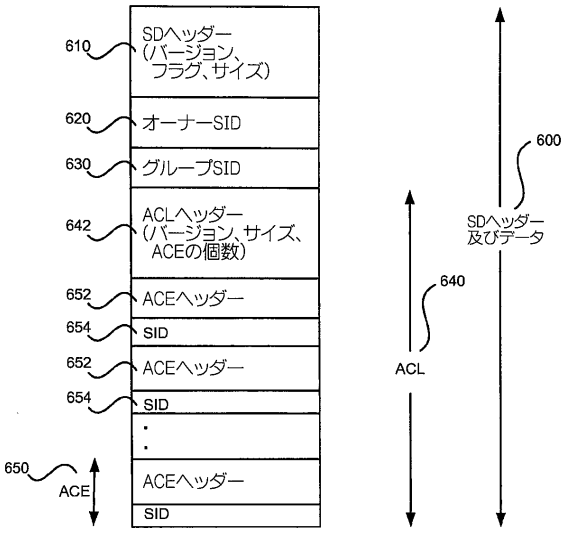
【図４】



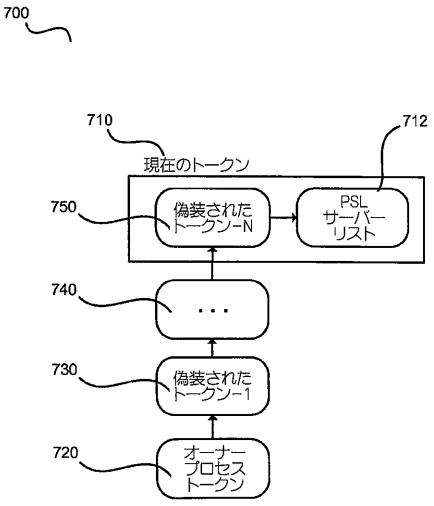
【図５】



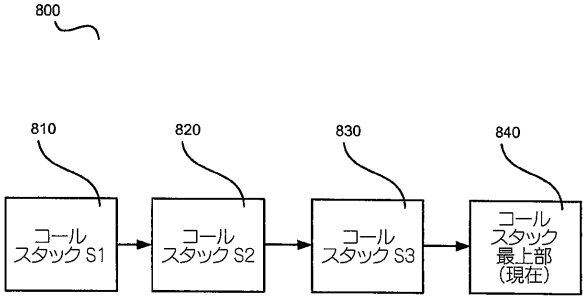
【図６】



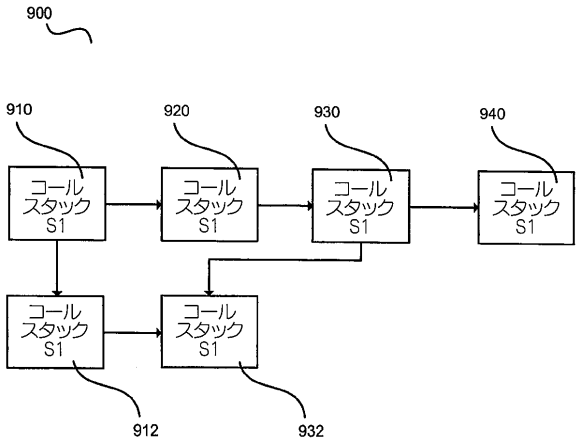
【図７】



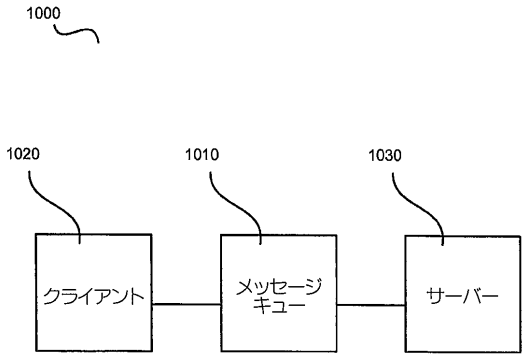
【図８】



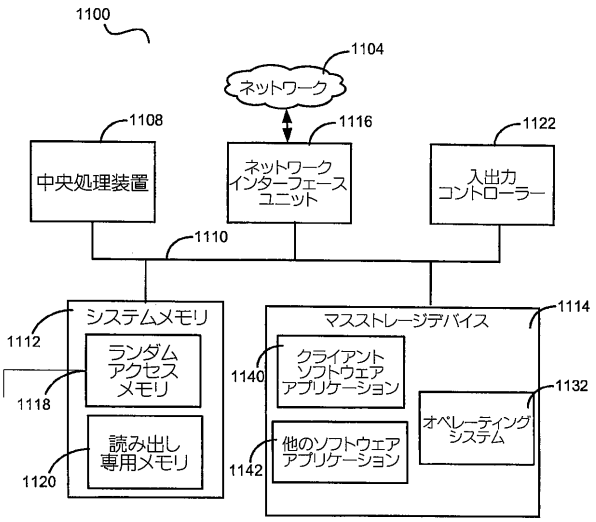
【図９】


900
910
920
930
940
コールスタックS1
コールスタックS1
コールスタックS1
コールスタックS1
コールスタックS1
コールスタックS1
912
932

【図１０】


1000
1020
1010
1030
クライアント
メッセージキュー
サーバー

【図１１】


1100
1104
ネットワーク
1108
1116
1122
中央処理装置
ネットワークインターフェースユニット
入出力コントローラー
1110
システムメモリ
1112
ランダムアクセスメモリ
1118
読み出し専用メモリ
1120
マスストレージデバイス
1114
1140
クライアントソフトウェアアプリケーション
1132
オペレーティングシステム
1142
他のソフトウェアアプリケーション

フロントページの続き

(72)発明者 コレス，ネイル・ローレンス
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 シェル，スコット・ランドール
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 サンダディ，アペンダー・レッディー
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 ヴァルス，アンジェロ・レナト
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 リオンス，マシュー・ジー
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 ジョルダン，クリストファー・ロス
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 ロジャース，アンドリュー
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 ゴパラン，ヤドヒュ
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 シエ，ボルーミーン
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ

審査官 戸島 弘詩

(56)参考文献 米国特許第０６３８５７２４（ＵＳ，Ｂ１）
特開２００５－１２９０６３（ＪＰ，Ａ）
米国特許第０６４１２０７０（ＵＳ，Ｂ１）
米国特許出願公開第２００６／０２５９９８０（ＵＳ，Ａ１）
特表２００２－５１７８５４（ＪＰ，Ａ）
米国特許第０５５８６２６０（ＵＳ，Ａ）
米国特許出願公開第２００７／００１１４５２（ＵＳ，Ａ１）
米国特許第０８３９７２９０（ＵＳ，Ｂ１）
特開平０６－２０２９８７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ２１／００－２１／８８