情報処理学会論文誌 Vol. 49 No. 9 3074–3084 (Sep. 2008)

GUIベースのコンピュータに適したアクセス制御ポリシ管理方法の提案

古市 実裕[†1] 工藤 道治[†2]

業務用コンピュータにおける機密情報の厳密な管理は，最も重要で緊急な課題の1つである．しかし，操作性や利便性を低下させずに既存のシステムにシームレスに統合できる現実的な解法が少ないため，セキュリティ事故がいつ起きてもおかしくない危険な状態が放置されている．本論文では，GUIベースのコンピュータに特化した柔軟なアクセス制御手法を提案する．本手法は，管理者が定めたルールに基づき，GUIやプロセスの状態に応じて，柔軟にアクセス制御ポリシを更新し，操作性や利便性を犠牲にせずに情報を保護する．広く業務に用いられているWindows PCにおいて，市販の商用アプリケーションを使った典型的な業務形態に適用する例を紹介し，提案手法の有効性を示す．

Access Control Policy Management for GUI-based Computer

Sanehiro Furuichi[†1] and Michiharu Kudo[†2]

Information flow control is one of the most important and urgent issues to be addressed in enterprise computing environment. However, there still remains many cases where the security incident could easily happen at any time due to poorly-managed sensitive information. This is because there are few practical solutions seamlessly integrated into existing computing system without damaging usability and operability. In this paper, we propose a context-aware access control framework in Windows-based computer, allowing administrators to enforce flexible and user-friendly access control policy according to contexts relevant to various user operations. We show the effectiveness of our approach by demonstrating the flexible access control in off-the-shelf application with a typical usage scenario using actual business applications.

1. はじめに

内部統制における重要な課題の1つに，いかに従業員にセキュリティポリシを順守させるかという問題がある．特に，機密情報のアクセス管理や情報フローの管理は，最も重要で緊急性の高い課題である．残念ながら，多くの組織において，日常業務で取り扱われる機密情報や顧客情報は，適切に管理されていない．多くの機密性の高い情報は，従業員の不適切な管理によって，PCの内部や外部メディアに散在しており，セキュリティ事故の起きやすい状態が放置されている．その原因として，日常業務の多くが，セキュリティ機能の乏しいWindowsベースの商用アプリケーションに依存しており，現状では決定的なソリューションが少ない点があげられる．

強制アクセス制御（MAC）機構を備えたセキュアOSは，この問題に対する1つの選択肢を与える．セキュアOSを使うと，管理者の設定に基づき，リソースアクセスを厳密に管理できる．近年，SELinux[1)]，AppArmor[2)]，LIDS[3)]など，Linuxベースのセキュア OSが普及し，企業内でも，Webサーバやデータベースなどのサーバ用途では広く使われている．

しかし，セキュアOSをGUIベースのクライアントPCで日常業務に使う場合，アクセス制御ポリシの粒度という観点で課題がある．たとえば，ワープロや表計算ソフトなどの大規模商用アプリケーションを使って，複数の文書を編集する場合を想定する．このとき，従来のプロセス単位の粗い制御では，ユーザが編集する文書の機密レベルに応じて，アクセス制御ポリシを動的に変更できない．本来は，編集中の文書の機密レベルに応じて，ポリシを柔軟に変更するのが望ましい．既存のセキュアOSのような画一的なポリシは，サーバ用途では問題ないが，GUIベースのクライアントPCでは使いにくく，結果的にユーザの負担につながる．ユーザのコンテキストに応じてきめ細かくアクセス制御ポリシを管理できる，クライアントPCに特化したアプローチが求められる．

本論文では，GUIベースのクライアントPCにおけるアクセス制御に焦点を当て，ユーザコンテキストに応じて柔軟にポリシを制御できる効率的な枠組みを提案する．商用アプリケーションをベースにした業務環境への適用を想定し，独自のポリシ強制手段を提供することで，不注意や故意による情報漏洩を未然に防ぐことを目的とする．提案手法をMicrosoft

†1 日本アイ・ビー・エム株式会社ソフトウェア開発研究所
Software Development Laboratory, IBM Japan, Ltd.
†2 日本アイ・ビー・エム株式会社東京基礎研究所
IBM Research, Tokyo Research Laboratory

Windows 2000，XP，Vista 上で実装し，既存環境にアドオンソフトとして追加できる実用的な設計にした．

提案手法は，3 章で述べるように，アプリケーションに依存しない高い汎用性を持つ．特に，4 章で詳しく述べるように，GUI の状態に応じて，迅速かつ柔軟にアクセス制御ポリシを更新できることを特徴とする．5 章では，実際の商用アプリケーションを用いたベンチマークテストで性能評価を行い，複雑なコンテキストを扱う細粒度なアクセス制御を施しても，ほとんどオーバヘッドがないことを示す．

2. 研究背景

本章では，GUI ベースのクライアント PC に求められる，アクセス制御ポリシの管理方法について議論する．クライアント PC 特有の複雑なコンテキストに対処し，セキュリティと利便性を両立させる理想的なアクセス制御に必要な要件を検討する．

2.1 強制アクセス制御

強制アクセス制御（MAC）は，情報アクセスや情報フローを管理するうえで，重要な要素技術の 1 つである．アプリケーションレベル，OS レベル，仮想マシンレベルなど，いくつかの実装形態が存在する．

たとえば，Enterprise Digital Rights Management システムは，アプリケーションレベルの MAC の 1 つと考えられる．Microsoft Windows Rights Management Services [4] や Adobe LiveCycle [5] は，代表的な例である．これらのシステムでは，アプリケーション自身が，文書ごとに定めた編集権限や印刷権限などのセキュリティポリシを解釈し，操作を制限する．残念ながら，このシステムで保護できるのは特定のアプリケーションに限られる．すべての業務アプリケーションが対応しない限り，業務全体を包括的に管理できない．また，特定ベンダに依存したポリシ管理機構の設計は，既存の IT システムとの統合を困難にし，コストに敏感な事業者には抵抗がある．

一方，SELinux [1]，AppArmor [2]，LIDS [3] などのセキュア OS は，情報フロー管理のための有力な選択肢である．OS の標準機能として MAC 機構を備えており，アプリケーションにまったく依存しない点も評価できる．しかし，アクセス制御の粒度という観点で見ると，既存のセキュア OS は，GUI ベースのクライアント PC で使うには，いささか柔軟性に欠ける．SELinux のドメイン遷移のように，ある程度，コンテキストに依存したポリシの適用は可能だが，GUI の状態などに応じて，稼働中のプロセスのポリシを柔軟に修正することは不可能であり，もう一歩踏み込んだ，きめ細かいポリシ管理が求められる．

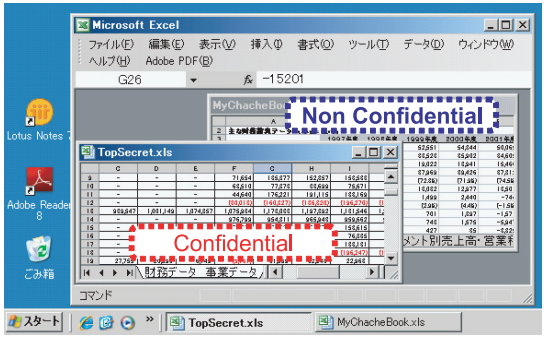


図 1　同一プロセス内で複数の文書を同時に表示する MDI アプリケーションの例
Fig. 1　An example of MDI application that opens multiple documents in a single process instance.

MAC 機構を備えた仮想マシンモニタの管理下で，仮想 OS 間の情報フローを制御する技術[6),7)] も存在する．仮想マシンモニタで限られた通信チャネルのみを監視すればよいので，単純な構成で済む．ただ，機密レベルごとに仮想環境を構築する必要があり，同じ OS やアプリケーションを複数の環境に導入するライセンスコストや，個々の仮想環境を管理する運用コストが課題となる．また，仮想 OS 間を頻繁に行き来する煩雑さも無視できない．仮想化技術に依存しなくても，プロセス間の情報フローさえ適切に制御できれば，十分実用的なアクセス制御が可能だと思われる．

2.2 GUI ベースのコンピュータに特有な課題

多くのクライアント PC は，直感的な操作が可能なウィンドウシステムを採用する．ユーザは，複数のアプリケーションのウィンドウを同時に表示しながら，マウスやキーボードを使って，ウィンドウを切り替えて作業する．残念ながら，セキュリティの観点から見ると，この利便性が管理をより困難にしており，ときには致命的なセキュリティ事故を招く．

たとえば，よく知られるクリップボードは，複数のアプリケーションの間でデータを自由に交換できる，便利な共有バッファである．あらゆるアプリケーションが自由にデータを書き込め，また，フォーマットさえ解釈できれば，何の制限もなくデータを読み込める．ただし，クリップボードは，カーネルオブジェクトとは異なり，十分なアクセス制御機構のないユーザオブジェクトである．そのため，同じデスクトップ内で動く不審なプログラムへの，意図せぬ情報漏洩を防げない．クリップボード以外にも，ウィンドウメッセージや画面スナップショット，プリンタなども，同様に注意を要するリソースである．

また，GUI の利便性がもたらす複雑なコンテキストを，いかにアクセス制御ポリシに反映させるかという課題も，クライアント PC 特有の問題である．たとえば，ワープロや表計算ソフトなどの大規模な商用アプリケーションは，1 つのプロセス内で複数の文書を同時に編集できる，MDI（Multiple Documents Interface）形式であることが多い．MDI 形式のアプリケーションでは，**図 1** に示すように，1 つのプロセス内で，機密レベルの異なる複数の文書を同時に編集できる．クリック 1 つでウィンドウを切り替えられ，プロセスを起動し直す必要がない．従来の古典的なアクセス制御では，プロセス単位の固定的なポリシしか適用できないため，このような複雑なアプリケーションには対応できない．安全策として最も厳しいポリシを過度に適用せざるをえず，ユーザの操作性が大きく低下する．本来は，作業内容に応じて，プロセスに適用するポリシを柔軟に変更するのが望ましい．

3. コンテキストアウェアなアクセス制御

2 章で述べたように，GUI ベースのクライアント PC では，アクセス制御ポリシにコンテキストを反映させる柔軟性が望まれる．本章では，GUI に起因するユーザコンテキストに応じて，様々な種類のリソースアクセスを柔軟に管理できる，新たな枠組みを提案する．提案手法は以下の特徴を持つ．

- リソースの種類やアプリケーションに依存しない汎用的で拡張性の高いアクセス制御を実現する．
- 主として GUI に起因するユーザコンテキストをアクセス制御ポリシに柔軟に反映し，操作性や利便性を損なわないアクセス制御を実現する．

以下，提案手法の設計思想をまとめる．

3.1 シームレスなポリシ強制方法

一般に，ソースコードが入手できない商用 OS では，フィルタドライバを用いて独自の機能拡張を行うことが多い[8),9)]が，カーネルオブジェクトに限定されるという課題がある．本研究では，Win32 API および COM インタフェースを対象にした Binary Interception[10),11)]をベースに，ユーザ空間における透過的なポリシ強制手法を検討した．

各プロセスに監視用の DLL を注入後，GDI32.dll，User32.dll，Kernel32.dll などに含まれる主要な Win32 API を監視し，ポリシに基づいてファイルやクリップボード，プリンタなどの低レベルリソースへのアクセスを制御する．また，Windows の主要機能が，分散オブジェクト[12)]の一実装形態である COM[13)]で提供されることに着目し，特定の COM インタフェースのメソッド呼び出しを監視し，高レベルリソースのアクセスを制御する．リソースの特性に応じて両手法を適宜使い分け，プロセスごとに個別に指定したポリシを適用する．さらに，API・COM の呼び出し履歴や，OS からアプリケーションへの通知メッセージを，ポリシを決定するためのイベントとしてフィードバックし，**図 2** に示すような，汎用的な枠組みを構築する．

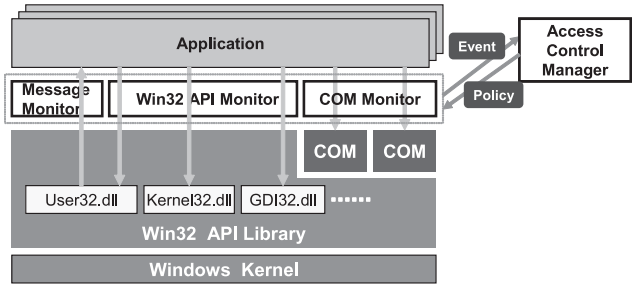


図 2　Win32 API，COM，メッセージの監視によるアプリケーション透過なアクセス制御機構
Fig. 2　Overview of our application-transparent policy enforcement mechanism by using Win32 API, COM and message interception.

表 1　実装システムで監視した Win32 API の詳細
Table 1　A detail of Win32 APIs monitored in the implemented system.

リソースタイプ	Win32 API	アクセス制御方法の詳細
ファイル	CreateFile*, DeleteFile,	ドライブ名やファイルパスに基づく Read/Write 制限
	CopyFile*, CopyFileEx*,	ドライブ属性やデバイス識別子に基づく Read/Write 制限
	MoveFile*, MoveFileEx*,	ファイルアクセスログの記録
	MoveFileWithProgress*	ファイルの複製や暗号化，ハッシュ値の記録
印刷	StartDoc*, EndDoc	プリンタ識別情報（機種名やホスト名など）に基づく印刷制限
	StartPage, EndPage	印刷ログの記録や用紙余白への識別情報埋め込み
クリップボード	SetClipboardData,	操作タイプやデータタイプに基づく制限
	GetClipboardData	別プロセスや別文書へのペースト制限，データ暗号化
プロセス	CreateProcess*	特定プロセス（特定ハッシュ値モジュール）の起動制限
	CreateProcessAsUser*	実行ユーザコンテキストの変更

* ANSI 版と Unicode 版の 2 種類が存在

表 1 および**表 2** に，実装したプロトタイプにおいて監視した Win32 API および COM インタフェースの一覧をまとめる．8 種類のファイル操作 API と COM の監視により，ファイルとして存在する不揮発な機密情報を保護し，4 種類の印刷 API と 2 種類のクリップボード API の監視により，メモリ上の揮発的な機密情報を保護する．情報漏洩経路としては，名

表 2　実装システムで監視した COM インタフェースの詳細
Table 2　A detail of COM interface monitored in the implemented system.

サービス名	IMAPI (CD Burning Service)
クラス	MSDiskMasterObj
インタフェース	IJolietDiscMaster
メソッド	AddData
アクセス制御方法の詳細	CD-R への Write 制限 操作ログの記録やファイル暗号化

前つきパイプやメールスロットなども考えられるが，アプリケーションへの依存性の高い非汎用的な経路である点と，未知のアプリケーションに対しては，プロセスの起動禁止やファイル自体の読み込み禁止で代替できるため，監視対象外とした．本実装は PC 内部での情報フロー管理に特化しており，PC 外部へのネットワークアクセスに関しては，Firewall などの他の手段を前提とする．

なお，不正な回避を防ぐため，ポリシを集中管理するアクセス制御マネージャの停止中には，一部の例外[⋆1]を除き，API 呼び出しを拒否するデフォルトポリシを適用する．また，要求したポリシや監視モジュールが適切に適用されているか確認するための API も提供する．

3.2　ユーザコンテキストの反映

GUI ベースのコンピュータでは，コンテキストに応じてきめ細かくアクセス制御を行わないと，ユーザの操作性にただちに影響を及ぼす．セキュリティ強化を理由にして，不必要にユーザの利便性を害するのは望ましくない．

情報漏洩の防止だけが目的なら，一律にデバイスを無効にしたり，専用のデスクトップ画面にアプリケーションを隔離したりする方法でもかまわない．しかし，通常，ユーザは，機密文書だけを扱うわけではない．メールの送受信や Web ページの閲覧など，多くの作業を同時に進める．また，当然，文書のセキュリティレベルに応じて適用すべき制限の種類も異なる．管理者指向の固定的なアクセス制御では，ユーザの利便性を大きく損ない，生産性が低下する．

我々は，ユーザに負担をかけない理想的なアクセス制御を実現するためには，ユーザのコンテキストを正しく理解し，それをリアルタイムにアクセス制御に反映する柔軟性が必要だと考える．提案手法は，以下の 2 つの特徴的なポリシ管理方法により，この課題を解決する．

- 個々のプロセス単位で，細粒度にアクセス制御ポリシを管理する．同一のプログラムを異なるポリシで別々に利用できる．
- ウィンドウやプロセスの状態を，アクセス制御ポリシにリアルタイムに反映する．ユーザの作業状況に応じて，柔軟なアクセス制御を実現する．

3.2.1　アクセス制御ポリシの適用粒度

既存のセキュア OS[1)–3)] のように，モジュール名に基づいてポリシを割り当てると，同じプログラムを異なる目的で同時に利用できない．たとえば，メモ帳（`Notepad.exe`）を 2 つ起動し，一方のプロセスで機密文書 `D1.txt` を編集し，もう一方のプロセスで通常文書 `D2.txt` を編集すると仮定する．このとき，機密文書 `D1.txt` の情報を保護するために，`Notepad.exe` に印刷制限やコピー制限などを施すと，本来，制限の必要がない文書 `D2.txt` も制限され，操作性が低下する．

この問題を解決するため，提案手法では，親プロセスがコンテキストに応じて，任意のポリシを適用した状態で子プロセスを起動する新たな API を提供する．ポリシが受け入れ可能な場合，Windows のプロセス起動 API を間接的に呼び出し，いったん，サスペンド状態で子プロセスを起動する．共有メモリ上のポリシ管理テーブルに，取得したプロセス ID とポリシパラメータを登録した後，サスペンド中のプロセスを再起動する．その後，新たに起動した子プロセスは，リソースアクセス API を呼び出すたびに，ポリシ管理テーブル上の該当するエントリを参照し，アクセス制御を行う．

この方式により，同じプログラムを目的に応じて任意のポリシで利用できる．機密文書と通常文書を同時に編集したり，重要度に応じて異なるポリシを割り当てたりすることが可能になる．また，共通のポリシを適用できる複数のプロセスをグループ化し，利便性を考慮した柔軟な情報フロー制御を行う．たとえば，同じグループのプロセス間のみコピー・ペーストを許可したり，片方向フロー制御，すなわち，別グループからのペースト（情報流入）は許可するが，別グループへのペースト（情報流出）は禁止したりする制御などが実現できる．

3.2.2　作業状態を反映した柔軟なアクセス制御

ウィンドウシステムでは，1 つのデスクトップ上で，多くのプログラムが同時に稼動し，ユーザの作業内容は刻々と変化する．このような環境で情報フローを適切に管理するためには，作業内容に応じた柔軟なアクセス制御が不可欠である．たとえば，MDI アプリケーションでは，編集文書のセキュリティレベルに応じて，適用するポリシを動的に変更すべきである．また，画面イメージを保護する場合には，ウィンドウが画面上に見えているか否かという点が重要な条件になる．機密文書が見えないときは，画面イメージの取得を制限する必要はない．

⋆1 たとえばシステムファイルの Read 操作は常時許可する．

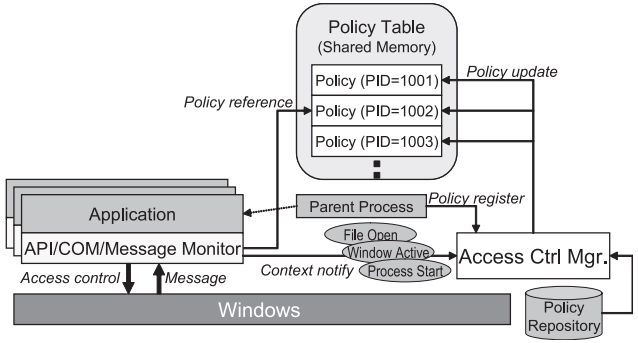


図 3　ユーザコンテキストの柔軟な反映を可能にする，GUI ベースのコンピュータのアクセス制御機構
Fig. 3　Architecture of our access control framework for GUI-based computer, allowing flexible access control reflecting user contexts.

このような作業内容を反映したアクセス制御を実現するため，提案手法では，**図 3** に示すように，ポリシを管理するアクセス制御マネージャに対して，GUI の状態やプロセスの稼動状態をフィードバックする仕組みを導入する．具体的には，特定の API 呼び出しや，OS からアプリケーションへのメッセージを，ポリシ更新に必要なイベントとしてフィードバックする．たとえば，ウィンドウのアクティブ状態や可視状態を判定するために，`WM_ACTIVATE`，`WM_CREATE`，`WM_DESTROY` などのウィンドウメッセージを監視する．ウィンドウクラス名やタイトル名，ウィンドウ属性などの情報から，どの文書が表示され，どの文書がアクティブになったか判定し，イベントとして通知する．イベントを通知されたアクセス制御マネージャは，状態変化に応じて共有メモリ上のアクセス制御ポリシを動的に更新する．ウィンドウの親子関係も考慮するので，ポップアップメッセージなどで一時的にフォーカスを失っても，正しく判定できる．またアプリケーションがプラグインをサポートする場合は，プラグインを導入してもよい．文書のフルパス名や URL など，汎用的なメッセージ監視より豊富な情報を入手できるので，より正確な判定が可能になる．実装したプロトタイプでは，主要な商用アプリケーションのプラグインを導入した．

3.3　アクセス制御モデル

以上に述べた提案手法の設計を，標準モデルと比較する．OASIS 標準のアクセス制御ポリシ記述言語仕様 XACML [14] では，アクセス制御モデルを，**図 4** のように定義する．ここで，PEP は *Policy Enforcement Point*，PDP は *Policy Decision Point* を意味する．ア

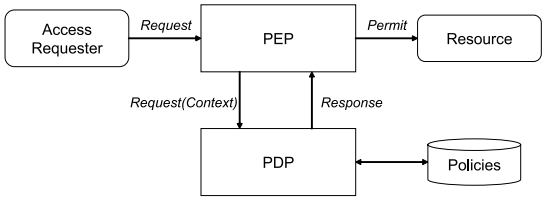


図 4　OASIS 標準で定義されるアクセス制御モデルの簡略図
Fig. 4　Conceptual diagram of access control model defined in OASIS standard.

クセス主体（*Access Requester*）が，リソースへのアクセスを要求すると，PEP はアクセスを実行すべきか否かを判断するため，アクセス主体やリソースの状態（*Context*）とともに，PDP に対して可否を問い合わせる．PDP は，ポリシレポジトリ内のポリシと照合し，アクセス要求を許可するか拒否するか判断する．PDP の判定結果に基づき，PEP はアクセス要求を実行するか，またはエラーを返す．

標準モデルに純粋に従えば，リソースアクセスが発生するたびに，コンテキストの抽出と，ポリシとの照合が必要になる．ただし，本研究が目指すように，GUI に依存した複雑なコンテキストをアクセス制御に反映させる場合，そのつどコンテキストの抽出とポリシの照合を行うのは，冗長で非効率である．

提案手法は，GUI の状態が変化したときに，事前に PDP に対するコンテキストの通知とポリシの再計算を行う．リソースアクセスが実際に発生したときには，再計算されたポリシを参照するだけで済む．このように，アクセス制御要求（Request）とコンテキスト通知を分離した点が，標準モデルに対する最も大きな拡張である．PEP は各プロセス内に分散化しており，リソースアクセスが発生した際に，コストの高いプロセス間通信や複雑なポリシの評価が必要ない．GUI ベースのコンピュータに適した実装形態になっている．

4.　アクセス制御ポリシの管理方法の詳細

本章では，アクセス制御ポリシの管理方法について，詳しく議論する．状態変化に対応可能なポリシの構成と，適用範囲の拡張について検討し，具体例を引用しながら，効率的にポリシを管理する方法をモデル化する．

4.1　状態変化に対応可能なポリシ構成

提案手法は，GUI やプロセスの状態を反映して，柔軟にアクセス制御をするため，**図 5** のように，アクセス制御ポリシを，適用条件ごとの個別の要素の集合体として扱う．親プロ

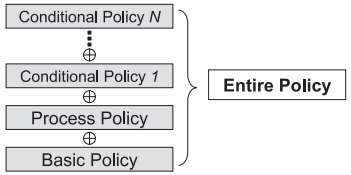


図 5　適用条件ごとの個別要素の組合せによるポリシ構築の概念図
Fig. 5　A structure of access control policy in our framework, which consists of multiple conditional policy elements.

セスから継承する基本ポリシ，プロセス稼働中に適用するプロセスポリシ，ウィンドウの可視状態やアクティブ状態に応じて適用する複数の状態依存ポリシを組み合わせ，最終的に対象プロセスに適用するポリシを構築する．

GUI やプロセスの状態が変化すると，その影響を受けるポリシ要素だけが部分的に追加または削除される．後述するように，他のプロセスから要求される一時的な制限も，状態依存ポリシの一種に分類され，各プロセスのポリシに動的に追加・削除される．

4.2　ポリシ適用範囲の拡張

ある特定のプロセスのみを対象にした狭いポリシしか記述できないと，GUI に起因する複雑なコンテキストには対処できない．他のプロセスにも影響を及ぼす広範な記述力が求められる．

たとえば，機密文書 `D1.txt` をメモ帳（`Notepad.exe`）で編集する場合を想定する．このとき，`Notepad.exe` における印刷や別名保存，クリップボードへのアクセスを制限するだけでは，`D1.txt` に含まれる機密情報を保護できない．他のすべてのプロセスに対して，機密文書 `D1.txt` へのアクセスや，画面スナップショットの取得を同時に制限する必要がある．

そこで，提案手法では，ポリシの適用対象を以下のように拡張する．

- **ローカルポリシ要素** ··· ある特定のプロセスに適用するポリシ要素
- **グローバルポリシ要素** ··· それ以外のすべてのプロセスに適用するポリシ要素

グローバルポリシ要素は，他のプロセスにとって，状態依存ポリシの一種であり，要求元プロセスの状態変化に応じて，随時追加または削除される．先の例では，**表 3** のポリシを適用すると，`Notepad.exe` からの情報漏洩と，他のプロセスによる情報の不正取得を，同時に防止できる．他のプロセスに適用した制限は，不要になれば即座に解除される．

このようにポリシの適用対象を拡張する概念は，従来の古典的な強制アクセス制御システムにはなく，個々のプロセスごとに個別に設定する必要があった．しかし，実際には，どのようなプロセスが同時に稼動するか，管理者は事前に把握できない．提案手法では，主体となるプロセスのポリシを，他のプロセスに追加的に反映できる仕組みを実現し，管理しやすいポリシの記述を可能にする．

表 3　Notepad.exe に適用するポリシの例
Table 3　An example policy for Notepad.exe.

適用対象	パラメータ
ローカルポリシ	印刷禁止，別名保存禁止
	クリップボードコピー禁止
グローバルポリシ	`D1.txt` へのアクセス禁止
	画面スナップショット禁止

4.3　効率的なポリシの再構築

提案手法の最大の特徴は，状況の変化に応じて，有効なポリシを迅速に再構築できる点にある．ここでは，ある親プロセスが，あるポリシで子プロセスを起動する場合を例にして，どのようにポリシを再構築するか述べる．

まず，子プロセスのポリシ P を 2 要素に分けて，

$$P = P_S + P_C$$

と表す．ここで，P_S は，子プロセス自身を含むすべてのプロセスの状態に依存しない，静的なポリシ要素を意味する．また P_C は，それらの状態に依存する動的なポリシ要素を意味する．加算はポリシの結合を意味し，たとえば，

$P_1 =$ *“Deny access to C drive”*,

$P_2 =$ *“Deny access to D drive”*,

とすると，

$P_1 + P_2 =$ *“Deny access to C or D drive”*.

という意味になる．

プロセスの状態に依存しない固定部分 P_S は，親プロセスから継承する基本ポリシ P_P と，子プロセス自身に要求するローカルプロセスポリシ P_L とに分けて，以下のように表現される．

$$P_S = P_P + P_L.$$

一方，可変部分 P_C は，1 つまたは複数の状態依存ポリシ要素で構成される．各状態依存ポリシ要素 C_L は，状態 c_i（$i = 1, \cdots, N$）の関数であり，適用条件 c_i が成立するときだけ適用される．すなわち，

$$C_L(c_i) = \begin{cases} Policy\ parameter & \text{if } c_i \text{ is satisfied,} \\ null & \text{Otherwise.} \end{cases}$$

さらに，稼働中の別のプロセス j が他のプロセスに要求するグローバルポリシ要素 G_j $(j = 1, \cdots, M)$ も，状態依存ポリシ要素の 1 つとして加算される．最終的に可変部分 P_C は，

$$P_C = \sum_{i=1}^{N} C_L(c_i) + \sum_{j=1}^{M} G_j$$

と表現される．ここで，N は条件の数，M は稼働中のプロセスの数を意味する．

ところで，子プロセスの稼働中に他のプロセスに制限を課す場合は，子プロセスのグローバルポリシ要素 P_G を，他のすべてのプロセスの既存ポリシに加算する．また，複数のグローバルな状態依存ポリシ要素 $C_G(c_i)$ を，条件 c_i が成立している間だけ加算する．

アクセス制御マネージャは，適用条件の変化がどのプロセスに影響を与えるか把握しており，GUI やプロセスの状態変化イベントを通知されるたびに，状態に関連付けされたポリシ要素を特定して，ポリシを迅速に再構築する．ある条件 c_i が成立しなくなった場合，子プロセスのポリシから，条件 c_i 成立時のローカルポリシ要素 $C_L(c_i)$ のみを除外する．また，他のプロセスに制限を課していた場合，各プロセスのポリシから，グローバルポリシ要素 $C_G(c_i)$ のみを除外する．

提案手法は，“*Deny Overriding*” の考え方に基づく．つまり，デフォルト無制限の状態をベースに，必要な制限を加算的に積み上げる．そのため，アクセス制御マネージャを除き，一般のプロセスは，制限を強化できても，緩めることはできない．

4.4　アクセス制御ポリシ再構築の具体例

提案手法の効果を示すため，具体的な利用シナリオを想定し，状態変化に応じてアクセス制御ポリシを更新する様子を説明する．

ここでは，Microsoft Word（`winword.exe`）を用いて，2 種類の文書を同時に編集する場面を想定する．一方の文書 “`secret.doc`” は，機密性の高い文書であり，情報を保護する必要がある．もう一方の文書 “`general.doc`” は，通常のビジネス文書であり，特に保護する必要性はない．ここで，Microsoft Word は MDI 形式のアプリケーションであり，2 つの文書を 1 つのプロセス内で同時に編集できる点に注意を要する．操作性や利便性を犠牲にせずに情報を保護するためには，様々なコンテキスト，すなわち，ユーザがどの文書にフォーカスを当てているか，あるいは，どの文書が画面上に見えているか，などの状態に応じて，

表 4　通常文書と機密文書を同時に編集するときの状態依存ポリシの例
Table 4　An example of conditional policy element while editing a normal document and a secret document simultaneously.

	状態	適用対象	ポリシ要素
P_1	`winword.exe` の稼働中	Local	—
		Global	`secret.doc` アクセス禁止
P_2	“`Secret.doc`” の表示中	Local	—
		Global	画面スナップショット禁止
P_3	“`Secret.doc`” のフォーカス中	Local	印刷禁止，別名保存禁止 クリップボードコピー禁止
		Global	—

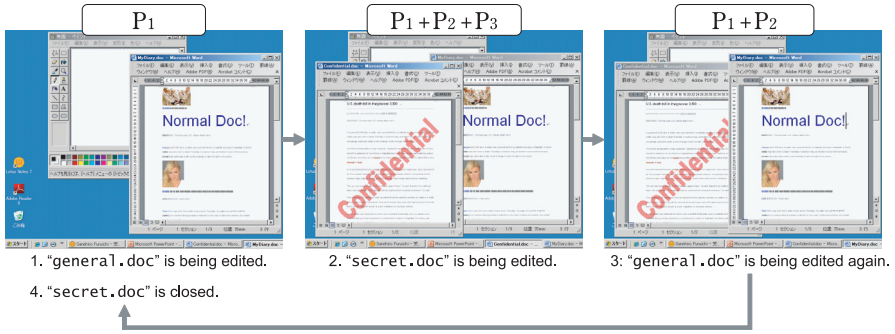


図 6　2 種類の文書を同時編集するときの GUI 状態の遷移と適用するポリシ
Fig. 6　Transition of GUI condition and policy while editing two documents.

プロセス `winword.exe` や他のプロセスに適用するポリシを動的に調整する必要がある．

ここでは，適用条件ごとに以下の 3 種類のポリシ要素を定義する．

- P_1: `winword.exe` が稼働している間だけ有効なポリシ要素
- P_2: `secret.doc` のウィンドウが表示されている間だけ有効なポリシ要素
- P_3: `secret.doc` のウィンドウがアクティブな間だけ有効なポリシ要素

P_1，P_2，P_3 の具体的なパラメータを，**表 4** に記す．**図 6** には，ユーザが以下の手順で操作した場合に，GUI の表示状態とポリシがどのように変化するかを示す．

(1)　最初に Microsoft Word を起動し，通常の文書 `general.doc` を開いて編集する．
(2)　続いて，同じ Word で，機密文書 `secret.doc` を開いて編集する．

(3)　再度，通常文書 general.doc のウィンドウを選択し，編集を継続する．
(4)　以上を繰り返した後，最後に機密文書 secret.doc を閉じる．

まず，通常の文書だけを編集しているときは，ポリシ P_1 のみが適用され，secret.doc へのアクセス以外には何の制限も課さない（**状態 1**）．

続いて，ユーザが機密文書 secret.doc を開くと，機密文書のウィンドウが表示されている間に有効なポリシ P_2 と，機密文書がアクティブな場合に有効なポリシ P_3 が加算される．この状態では，Word には最も厳しいポリシが適用され，印刷もクリップボード操作も別名保存もできない．また，画像による情報漏洩を防ぐため，稼動中の他のプロセスは，画面スナップショットの取得を制限される（**状態 2**）．

次に，機密文書 secret.doc を開いたまま，通常文書 general.doc のウィンドウにフォーカスを切り替えると，ポリシ P_3 が解除される．その結果，Word では，それまで制限されていたクリップボードの操作や印刷，別名保存が可能になる．ただし，画面上には機密文書 secret.doc のウィンドウが表示されたままなので，ポリシ P_2 は適用され続ける（**状態 3**）．

最後に，ユーザが機密文書を閉じるか，Word を終了すると，すべての関連する制限がいっせいに解除される（**状態 4**）．

以上の各状態におけるポリシを**表 5** にまとめる．この例では，文書の性質にかかわらず，全操作を一律に禁止する厳しいポリシを適用したが，実際の運用では，文書の機密レベルや重要度に応じて，個別に異なるポリシを適用したり，同一機密レベルの文書間やプロセス間では，コピー・ペーストを許可したりするなど，柔軟な運用が可能である．これらのポリシは，管理者により集中管理され，ファイルサーバやデータベースから文書ファイルをダウンロードする際に自動的に登録される．

表 5　GUI の各状態に対する適用ポリシの一覧
Table 5　A list of effective policy for each GUI condition.

	状態 1	状態 2	状態 3	状態 4
適用ポリシ	P_1	$P_1+P_2+P_3$	P_1+P_2	P_1
Word				
印刷	許可	拒否	許可	許可
コピー	許可	拒否	許可	許可
別名保存	許可	拒否	許可	許可
Others				
画面スナップ	許可	拒否	拒否	許可
機密文書読込	拒否	拒否	拒否	拒否

5. 評価と考察

GUI の操作に応じて頻繁にアクセス制御ポリシを更新すると，性能低下が懸念されるが，提案手法では，コンテキストの通知とポリシの判定を分離することにより，オーバヘッドを最小限に抑制できる．そこで，GUI アプリケーションを用いたベンチマークテスト，BAPCO SYSmark 2004 SE [15] で性能評価を行った．

SYSmark 2004 は，オフィス業務やコンテンツ作成業務の生産性を評価するため，PC ベンダやソフトウェアベンダを中心に作成した業界標準のベンチマークであり，**表 6** に示した複数のアプリケーションを同時に使用したときの，個々のタスクの応答時間を評価指標とする．ベンチマークスコアは，参照マシンと呼ぶ標準的仕様の PC における応答時間に対する相対値として算出される*1．従来のベンチマークテストが，CPU 演算やメモリ・ディスク I/O など，特定デバイスの局所的な性能評価を目的とするのに対し，SYSmark 2004 は，

表 6　BAPCO SYSmark 2004 SE で用いるアプリケーション
Table 6　A list of applications used in BAPCO SYSmark 2004 SE.

Application Name	Category
Adobe Acrobat 5.0.5	I
Adobe After Effect 5.5	I
Adobe Photoshop 7.01	I
Adobe Premiere 6.5	I
Dicscreet 3D Studio Max 5.1	I
Macromedia Dreamweaver MX	I
Macromedia Flash MX	I
Microsoft Access 2002	O
Microsoft Excel 2002	O
Microsoft Internet Explorer 6	O
Microsoft Outlook 2002	O
Microsoft PowerPoint 2002	O
Microsoft Word 2002	O
Microsoft Windows Media Encoder 9	O
Network Associates McAfee VirusScan 7.0	I/O
ScanSoft Dragon NaturallySpeaking 6	O
Nico Mark Computing WinZip 8.1	I/O

I: Internet Content Creation, O: Office Productivity

*1 たとえば，スコア 100 は，参照マシンと同じ応答時間，スコア 200 は，参照マシンに対して応答時間が半分であることを意味する．

表 7　ベンチマークテストの実行環境仕様
Table 7　A specification of the test environment.

OS	Microsoft Windows XP SP2
CPU	Intel Pentium M 2.00 GHz
RAM	1024 MB
Video	ATI Mobility Fire GL T2 (128 MB)
Display	1600 × 1200 × 32 bpp

表 8　BAPCO SYSmark 2004 SE のスコア
Table 8　Score of BAPCO SYSmark 2004 SE.

Test Category	無効	有効	性能低下
Internet Content Creation			
3D Creation	150	139	8%
2D Creation	200	196	2%
Web Publication	124	121	2%
Overall	155	149	4%
Office Productivity			
Communication	53	52	2%
Document Creation	97	96	1%
Data Analysis	81	79	3%
Overall	75	73	3%
Total	**108**	**106**	**2%**

GUI アプリケーションの体感性能を総合的に評価することを目的としており，提案手法を評価するベンチマークとして適切であると考える．

表 7 に，テスト機の仕様を記す．**表 8** には，提案したアクセス制御機構を有効にした場合と，無効にした場合の，ベンチマークスコア，および性能低下の割合を示す．提案手法による細粒度なアクセス制御を実施しても，平均で 2%，最大でも 8%の性能低下しか認められず，実用的には影響がないことを確認できた．

4 章で紹介したシナリオは，ウィンドウの可視状態とアクティブ状態，プロセスの稼動状態の 3 つの条件しか反映していないが，それでも実用的には十分有用なアクセス制御ができる．提案手法は，状態の種類に依存しない汎用的な枠組みを提供するので，メニューやダイアログボックスを操作したときなど，より詳細なコンテキストに応じて，きめ細かいアクセス制御もできる．

6. 関 連 研 究

提案手法のように，Binary Interception によるセキュリティ強化や機能拡張を行う先行研究は数多く存在する[16)–21)]．その多くは，既存の OS に，いかにして新たなポリシ強制ポイント（PEP）を追加するかに焦点を当てており，主にファイルやネットワークなどのカーネルオブジェクトを対象にしたアクセス制御方法に特化している．それに対して，本研究では，ポリシ強制手段の 1 つとして Binary Interception を利用しているが，むしろ，研究の焦点は，アクセス制御ポリシの柔軟できめ細かい管理方法に置く．とりわけ，GUI ベースのコンピュータにおける状況依存性に注目する点が，従来研究と方向を異にする．

一方，セキュリティ機構に Binary Interception を用いることの是非は，長い間議論の対象であった．近年，時間軸上のアトミック性の不備をついた Binary Interception の脆弱性や危険性が指摘されている[9),22),23)]．我々は，現時点では，このような脆弱性のあるセキュリティ機構を導入するか否かは，それを運用する組織自身が判断すべきだと考える．操作性や管理のしやすさ，互換性や投資効果などを総合的に考慮し，脆弱性に起因するリスクを理解したうえで最終的に判断すべきである．ただ，近い将来，OS 本体にコンテキスト依存のアクセス制御機構が取り込まれるときが来れば，課題は解決されると信じる．

ウィンドウシステムのアクセス制御に関する研究もさかんである．ウィンドウシステムや OS カーネルに手を加えやすいため，多くは X Window System を対象にしている．文献 24) や 25) では，SELinux の強制アクセス制御を X Window System に拡張する方法が述べられている．Shapiro ら[26)] は，専用 OS 上のセキュア GUI として，EROS Trusted Window System を提案した．WindowBox[27)] は，Windows を対象にした数少ない研究の 1 つであり，特定のアプリケーションを専用のデスクトップ画面に隔離して，情報を保護する．これらの研究はいずれも，情報の機密性や完全性の確保というセキュリティ要件に主眼を置く．一方，本研究は，日常業務への適用を念頭に置き，実用性や柔軟性とのバランスを考慮する点が異なる．

コンテキストアウェアなアクセス制御という観点でも，多くの先行研究が存在する．文献 28) は，プライバシ情報に対するアクセス制御ポリシを，インターネット接続状況などに応じて動的に管理する方式を提案している．また，文献 29) は，位置情報に基づいて状況依存のアクセス制御を行う SBAC（Situation Based Access Control）を提案しており，ウェアラブルコンピュータにおける柔軟なサービス環境を実現する．文献 30) は，場所などの環境に応じてロールを判断してアクセス制御を行う Environment RBAC を提案してい

る．本研究とこれらの研究は，状況依存のアクセス制御を行う点で，共通の目的を持つが，本研究では，コンピュータの GUI の表示状態に応じて，効率良く状況依存のアクセス制御を行う方法に特化しており，その点で従来の研究と異なる．

7. ま と め

本論文では，GUI ベースのコンピュータにおいて，ユーザの操作性を可能な限り犠牲にしないために，コンテキストに応じて柔軟にアクセス制御を行う枠組みを提案した．提案手法は，アプリケーションに依存しない高い汎用性を持ち，既存の IT インフラにもシームレスに統合できる．性能評価の結果，GUI に起因する複雑な状況依存アクセス制御を行っても，実用上問題ないことを確認できた．

本提案手法の特徴である，プロセス単位の細粒度なアクセス制御や，GUI の状態を反映させた柔軟なアクセス制御により，ユーザの操作性や作業効率に影響を与えずに，ポリシに基づいて適切に情報フローを管理できる．セキュリティのみでなく，操作性や利便性にも配慮した，より現実的なセキュリティ製品の提供が可能になる．

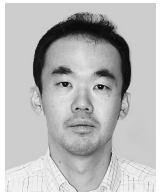
今後の展望として，パイプなどの他のプロセス間通信への拡張，Windows 以外のプラットフォームへの展開，既存インフラとの親和性の高いポリシ記述言語の検討などを進める．

参 考 文 献

1) Loscocco, P. and Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System, *Proc. FREENIX Track: 2001 USENIX Annual Technical Conference* (2001).
2) Bauer, M.: Paranoid Penguin: An Introduction to Novell AppArmor, *Linux Journal*, Vol.2006, No.148, p.13 (2006).
3) LIDS Project. http://www.lids.org
4) Microsoft Corporation: Technical Overview of Windows Rights Management Services for Windows Server 2003 (2003).
5) Adobe Systems, Inc.: LiveCycle Policy Server. http://www.adobe.com/products/server/policy/
6) Hitachi Software Engineering: Data Trans Guardian. http://www.hitachi-sk.co.jp/products/secure-linux/solution/2win
7) Griffin, J.L., Jaeger, T., Perez, R., Sailer, R., van Doorn, L. and Caceres, R.: Trusted Virtual Domains: Toward Secure Distributed Services, *Proc. 1st IEEE Workshop on Hot Topics in System Dependability* (2005).
8) Microsoft Corporation: Filter Driver Development Guide 1.0a (2004).
9) Watson, R., Feldman, B., Migus, A. and Vance, C.: Design and Implementation of the TrustedBSD MAC Framework, *3rd DARPA Information Survivability Conference and Exhibition* (*DISCEX3*) (2003).
10) Hunt, G. and Brubacher, D.: Detours: Binary Interception of Win32 Functions, *Proc. 3rd USENIX Windows NT Symposium*, pp.135–144 (1999).
11) Hunt, G. and Scott, M.L.: Intercepting and Instrumenting COM Applications, *Proc. 5th Conference on Object-Oriented Technologies and Systems* (*COOTS '99*), pp.45–56 (1999).
12) Szyperski, C.: *Component Software: Beyond Object-Oriented Programming*, ACM Press (1999).
13) Microsoft Corporation and Digital Equipment Corporation: The Component Object Model Specification (1995).
14) Moses, T.: eXtensible Access Control Markup Language (XACML) Version 2.0, Technical report, OASIS Standard (2005).
15) BAPCO: An overview of SYSmark 2004 SE (2005). http://www.bapco.com
16) Ghormley, D.P., Petrou, D., Rodrigues, S.H. and Anderson, T.E.: SLIC: An Extensibility System for Commodity Operating Systems, *Proc. USENIX Annual Technical Conference* (*NO 98*) (1998).
17) Acharya, A. and Raje, M.: MAPbox: Using Parameterized Behavior Classes to Confine Applications, *Proc. 2000 USENIX Security Symposium* (2000).
18) Fraser, T., Badger, L. and Feldman, M.: Hardening COTS Software with Generic Software Wrappers, *IEEE Symposium on Security and Privacy*, pp.2–16 (1999).
19) Fraser, T.: LOMAC: Low Water-Mark Integrity Protection for COTS Environments, *IEEE Symposium on Security and Privacy*, pp.230–245 (2000).
20) Kato, K. and Oyama, Y.: SoftwarePot: An Encapsulated Transferable File System for Secure Software Circulation, *Proc. International Symposium on Software Security*, LNCS-2609, pp.112–132 (2003).
21) Goldberg, I., Wagner, D., Thomas, R. and Brewer, E.: A Secure Environment for Untrusted Helper Applications: Confining the Wily Hacker, *Proc. 6th USENIX Security Symposium*, pp.1–13 (1996).
22) Watson, R.N.M.: Exploiting Concurrency Vulnerabilities in System Call Wrappers, *1st USENIX Workshop on Offensive Technologies* (*WOOT '07*) (2007).
23) Garfinkel, T.: Traps and Pitfalls: Practical Problems in System Call Interposition Based Security Tools, *Proc. Network and Distributed Systems Security Symposium* (*NDSS 2003*) (2003).
24) Kilpatrick, D., Salamon, W. and Vance, C.: Securing The X Window System with SELinux, Technical Report #03-006, NAI Labs (2003).
25) Walsh, E.F.: Application of the Flask Architecture to the X Window System

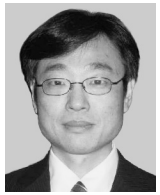
Server, *Proc. 2007 SELinux Symposium* (2007).
26) Shapiro, J.S., Vanderburgh, J., Northup, E. and Chizmadia, D.: Design of the EROS Trusted Window System, *Proc. 13th USENIX Security Symposium* (2004).
27) Balfanz, D. and Simon, D.R.: WindowBox: A Simple Security Model for the Connected Desktop, *Proc. 4th USENIX Windows Systems Symposium* (2000).
28) Suzuki, K., Mouri, K. and Okubo, E.: Salvia: A Privacy-Aware Operating System for Prevention of Data Leakage, *International Workshop on Security* (*IWSEC2007*) (2007).
29) 宮前雅一，寺田　努，塚本昌彦，西尾章治郎：ウェアラブルコンピューティング環境のための状況依存アクセス制御機構，電子情報通信学会和文論文誌 D，Vol.J88-D1, No.3, pp.617–628 (2005).
30) Covington, M.J., Long, W., Srinivasan, S., Dev, A.K., Ahamad, M. and Abowd, G.D.: Securing Context-Aware Applications Using Environment Roles, *Proc. 6th ACM Symposium on Access Control Models and Technologies*, pp.10–20 (2001).

(平成 19 年 11 月 30 日受付)
(平成 20 年 6 月 3 日採録)



古市　実裕（正会員）

1994 年東京大学工学部計数工学科卒業．1996 年同大学大学院工学系研究科修士課程修了．同年日本アイ・ビー・エム株式会社東京基礎研究所入社．2008 年より同ソフトウェア開発研究所に所属．コンピュータ・アーキテクチャ，システム・ソフトウェア，ユーザ・インタフェース，情報セキュリティ等の研究に従事．電子情報通信学会会員．



工藤　道治（正会員）

1988 年東京大学大学院工学系研究科修士課程修了，同年日本アイ・ビー・エム株式会社東京基礎研究所入社．情報セキュリティの研究・開発に携わる．主にアクセス制御・セキュリティポリシの研究に従事．2001 年に米国標準化団体 OASIS において XACML 委員会を設立，2003 年 2 月に国際標準になる．2002 年東京大学より博士（工学）の学位授与．現在，東京基礎研究所システム管理&コンプライアンス部門のシニアリサーチャー．電子情報通信学会会員．