情報処理学会論文誌 Vol. 49 No. 7 2656–2666 (July 2008)

推薦論文

アプリケーションレベル分散セキュアグループのための Chinese-Wall 型プロセス隔離手法

勝野恭治[†1,†2] 渡邊裕治[†1]
古市実裕[†3] 工藤道治[†1]

分散セキュアグループは，分散したノード上のコンポーネント間で構築したグループに対してセキュリティポリシを強制適応できる分散コンピューティングプラットフォームである．現在提案されているプロトタイプの主流はコンポーネント間の強固な隔離を実現するために仮想マシンモニタが用いられているため，アプリケーションの運用や管理が複雑になり，利便性が低下する恐れがある．そこで，プログラムリファレンスモニタを用いて利便性への影響を軽減するアプリケーションレベル分散セキュアグループが期待されている．さらに既存環境への影響を最小限に抑えるために，既存 OS やアプリケーションのコードを変更しない実現方法が望ましい．本論文では，既存環境への影響を最小限に抑えたアプリケーションレベル分散セキュアグループを実現する，Chinese-Wall ポリシの概念を利用したプロセス隔離手法 CWPC（Chinese-Wall Process Confinement）を提案する．CWPC は既存 OS やアプリケーションのコードを変更しないで実現するため既存環境への影響を最小限に抑えることができる．さらに本論文では，既存環境に対する変更を最小限に抑えられることを示すために，幅広く用いられている Microsoft Windows 環境，およびその上で頻繁に利用されるオフィスアプリケーションを検証対象として，ドキュメントの安全な取扱いを実現するプロトタイプシステムを実装し，性能評価を通して，本論文のアプローチが実用的であることを示す．

Chinese-wall Process Confinement for Application-level Distributed Coalitions

Yasuharu Katsuno,[†1,†2] Yuji Watanabe,[†1]
Sanehiro Furuichi[†3] and Michiharu Kudo[†1]

A distributed coalition is an attractive distributed computing platform to support distributed mandatory access controls for resources whose security policies differ for each group of components over nodes. Some prototypes of distributed coalitions have been implemented with a virtual machine monitor for strong confinement. But, this approach causes complicated operations and managements for applications. Then, an application-level distribution coalition using a program-transparent reference monitor with minimum impact on the usability is exptected. In this paper, we propose a Chinese-Wall Process Confinement (CWPC) to build an application-level distributed coalition. CWPC is a new policy-based process confinement mechanism that uses an effective but simplified variant of the Chinese-Wall policy security model without changing the code of the OS or applications. We implement a prototype system for office applications on Microsoft Windows platform, evaluate its overheads, and show that our approach is acceptable.

1. はじめに

1.1 背　　景

分散セキュアグループ（Distributed Coalition）は，複数のノード上のコンポーネント間でドメインと呼ばれるグループを構築し，各ドメインに対してそれぞれ異なるセキュリティポリシを強制適応させることができる分散コンピューティング・プラットフォームである．分散セキュアグループの代表的な利用ケースとして次の 3 つが考えられている．

- ドキュメントの安全な取扱い
  ドメインを競合会社ごとに構築して，不適切な情報混在を発生させずに，複数の競合会社のドキュメントを安全に取り扱うことができる．
- グリッドコンピューティングにおける計算タスクの安全な分散実行
  ノードの物理的な制約に縛られない仮想的な隔離実行環境としてドメインを構築して，利害関係の異なる要求者からの処理要求を同時に扱うことができる．

†1 日本アイ・ビー・エム株式会社東京基礎研究所
IBM Research, Tokyo Research Laboratory
†2 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†3 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.
本論文の内容は 2007 年 7 月のマルチメディア，分散，協調とモバイル（DICOMO2007）シンポジウムにて報告され，マルチメディア通信と分散処理研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

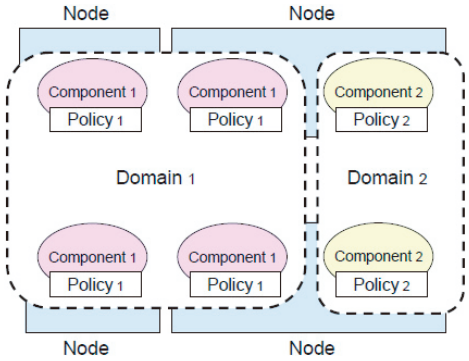


図 1　分散セキュアグループ
Fig. 1　Distributed coalitions.

- 企業間・部門間のデータの安全な共有
  近年の動的な企業連携・合併に対応して，企業や部門で取り扱うデータを仮想的なドメインで管理して，企業間・部門間における柔軟なデータ共有を実現する．

図 1 は分散セキュアグループの一例を示している．$Domain_1$ と $Domain_2$ という 2 つのドメインがドメイン管理者によって構築されている．コンポーネント $Components_1$ と $Components_2$ はそれぞれ $Domain_1$ と $Domain_2$ に参加している．ドメイン管理者は，ファイルといったリソースに対するアクセス制御を行うポリシをドメインごとに定義する．コンポーネントはドメインに参加している間はドメインで定義されているポリシを強制適応されなければならない．図中では，$Domain_1$ に対して $Policy_1$，$Domain_2$ に対して $Policy_2$ がそれぞれ定義されていて，$Components_1$ は $Policy_1$ を，$Components_2$ は $Policy_2$ を強制適応されている．

分散セキュアグループでは，各ノード上でアプリケーションサービスを提供するアプリケーション・コンポーネントが動作している．ノード上のアプリケーション・コンポーネントがそれぞれ異なるドメインに参加できるようにするために，各コンポーネントは隔離されて別々に実行され，コンポーネントの動作が他のコンポーネントに影響を与えないようにする必要がある．そこで，セキュア・インフラストラクチャと呼ばれるコンポーネントが同一ノード上のアプリケーション・コンポーネントを隔離して，参加しているドメインで定義されているリソースアクセスに対するポリシを各アプリケーション・コンポーネントに対して強制適応する役割を担う．セキュア・インフラストラクチャには次のような実現方法がある．

- 仮想マシンモニタの場合：アプリケーション・コンポーネントは仮想マシン，リソースはディスクボリュームやネットワークインタフェース
- プログラムリファレンスモニタの場合：アプリケーション・コンポーネントはアプリケーション・インスタンス，リソースはファイルやソケット

現在，Terra[1]，NetTop[2]，Shamon[3]，European Multilaterally Secure Computing Base（EMSCB）[4] など，いくつかのプロジェクトが分散セキュアグループのプロトタイプを提案している．それらの主流は，アプリケーション・コンポーネントの強固な隔離[5] を実現するために，仮想マシンモニタを用いてセキュア・インフラストラクチャを実現している．しかし，仮想マシンモニタを利用したセキュア・インフラストラクチャの実現は，リソースに対するアクセス制御が疎粒度になるため，利便性の低下をもたらす．

同一アプリケーションを複数のドメインに同時に参加させることを考える．複数ドメインへの同時参加を実現するためには，アプリケーションがインストールされた仮想マシンを事前に参加させるドメインと同じ数用意して，仮想マシンを複数同時に動かす必要がある．よって，アプリケーションの運用や管理が複雑になる．またドメインと同数のソフトウェアライセンスが必要になる可能性がある．さらに，複数仮想マシンの同時実行は，ノードに要求するハードウェアリソース，特に物理メモリのサイズが高くなるという問題を生じさせる．

そのため，プログラムリファレンスモニタによる細粒度なリソースアクセス制御を実現して利便性への影響を最小限にする，アプリケーションレベル分散セキュアグループの実現が期待されている．さらに既存環境への影響を最小限に抑えるために，既存 OS やアプリケーションのコードを変更しない実現方法が望ましい．

1.2　本論文の貢献

本論文では，既存環境への影響を最小限に抑えたアプリケーションレベル分散セキュアグループを実現するための Chinese-Wall 型プロセス隔離手法 CWPC（Chinese-Wall Process Confinement）を提案する．CWPC は，OS プロセスに対する強制アクセス制御機構上に実現される，隔離を保証するポリシとして広く使われている Chinese-Wall ポリシ[6] の概念を利用したプロセス隔離技術である．ドメインを管理するドメイン管理者がドメインごとにセキュリティラベル[7] を定義して，ドメインで管理するアプリケーションのインスタンスとファイルやクリップボード，ソケットといった OS レベルのシステムリソースに添付する．セキュリティラベルが添付されたインスタンスは同一ラベルが添付されたリソースのアクセスのみを許可される．以上のプロセスのリソースアクセス制御を行うことで，各プロセスの

実行を隔離する．CWPC は既存 OS やアプリケーションのコードを変更しないで実現されるため既存環境への影響を最小限に抑えることができる．

本論文では提案手法が，既存環境に対する変更を最小限に抑えて適用できることを示すために，一般に普及する Microsoft Windows 環境，およびその上で頻繁に利用されるオフィスアプリケーションを検証対象として選定する．そして，Microsoft Windows 上のオフィスアプリケーションによる安全なドキュメント取扱いを実現するプロトタイプシステムを実装し，性能評価を行うことで，本論文のアプローチが実用的であることを示す．

本論文の構成は次のとおりである．2 章で設計，3 章でプロトタイプ実装について述べる．4 章でプロトタイプシステムの性能評価を行う．関連研究について 5 章で議論する．最後に，6 章で結論と今後の課題について述べる．

2. 設　　計

2.1 アーキテクチャ

アーキテクチャを**図 2** に示す．ノードは，アプリケーション・コンポーネント（Application Component，AC），コンポーネント・コントロール・エージェント（Component Control Agent，CCA），セキュア・インフラストラクチャ（Secure Infrastructure，SI）の 3 種類のコンポーネントで構成される．AC はアプリケーションのインスタンス（OS プロセス）である．CCA は各アプリケーション・コンポーネントに添付して，アクセス制御ポリシに

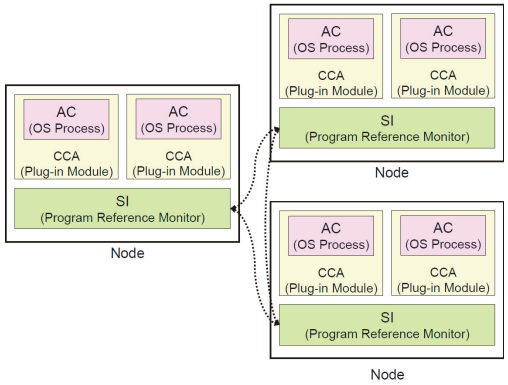


図 2　システムアーキテクチャ
Fig. 2　System architecture.

基づいてファイルやクリップボード，ソケットといったリソースに対する強制アクセス制御を行うプラグイン・モジュールである．SI はプログラムリファレンスモニタをベースにして，CCA と協調して AC の隔離を行い，AC の実行が他の AC に影響を与えないようにする．分散セキュアグループへの参加を希望するすべてのノード上で SI と CCA が動作している必要がある．各ノード上の SI は同一ノード上の CCA と協調しつつ，他のノード上の SI と協業することによって複数ノードにまたがってドメインを構築する．

ドメインの管理を行うドメイン管理者はドメインごとにセキュリティラベル[7] を定義して，利益相反の関係にあるセキュリティラベルの組を利益相反グループとして定義する．そして，各ドメインに所属する AC とリソースにラベルを添付する．セキュリティラベルが添付されたアプリケーション・コンポーネントは同一ラベルが添付されたリソースのアクセスを許可される．これによってドメイン単位のアクセス制御を実現する．

本論文では，分散セキュアグループに参加するすべてのノード上で SI と CCA が正しくインストールされており，すべての AC に対して CCA が添付し，SI と CCA が正常に動作していることを仮定している．そのため，ルートキットやマルウェアなどによる SI や CCA への攻撃は想定していない．

2.2 情報フロー制御の要件

情報フロー制御[8] の要件を以下に示す．

- 同一ラベルが添付されているリソース間で情報が行き来することを許可する．
- 同一の利益相反グループに属していないラベルが添付されているリソース間で情報が行き来することを許可する．
- ラベルが添付されていないリソース間で情報が行き来することを許可する．
- 同一の利益相反グループに属しているラベルが添付されているリソース間で情報が行き来することを禁止する．
- ラベルが添付されているリソースと添付されていないリソースとの間で情報が行き来することを禁止する．

本論文ではリソースへの強制アクセス制御とプロセス隔離という 2 つの機能を用いて，これらの要件を満たす．

2.2.1 リソースへの強制アクセス制御

プロセスによるリソースアクセスを制御する方法として，リソースにアクセスするデバイスドライバにフィルタドライバを挿入するカーネル空間アプローチ[9)–13)] と，プロセスがリソースにアクセスする際に必ず利用される基本 API をフックするユーザ空間アプロー

チ[14)–16)] の 2 種類がある．対象となる環境や利用シナリオに応じて適切なアプローチが選択される．

カーネル空間アプローチもユーザ空間アプローチも既存コードに対してプラグインモジュールを挿入する方法をとるため，どちらのアプローチを選択しても既存 OS やアプリケーションのコードを変更しないため，既存環境への影響を最小限に抑えることができる．

2.2.2 プロセス隔離

本論文で提案する CWPC は，OS プロセスに対するリソースへの強制アクセス制御技術と連携して，Chinese-Wall ポリシ[6)] に基づいて各プロセスの実行を隔離して，プロセス実行が他のプロセス実行に影響を与えないことを保証する．Chinese-Wall ポリシは，利害相反が起こるアクセスを安全に制御するポリシとして広く使われている．Chinese-Wall ポリシでは，セキュリティラベルと，利益相反の関係にあるセキュリティラベルの組を定義する．そして，実行主体であるサブジェクトと実行主体がアクセスするリソースであるオブジェクトそれぞれにラベルを付与して，サブジェクトとオブジェクトそれぞれに添付されているラベルが利益相反関係であればアクセスを拒否し，利益相反の関係ではなければアクセスを許可する．

CWPC は，Chinese-Wall ポリシの概念を利用してプロセスのリソースアクセスを制御することによって，各プロセスの実行を隔離を行う．まず AC を，ドメインに参加できるラベル対象 AC（Label-attachable AC）と，参加できないラベル対象外 AC（Label-free AC）に分類する．そして，ラベル対象 AC をサブジェクトにして，オブジェクトをファイルやクリップボード，ソケットといったノード上のリソースにする．セキュリティラベルは各ドメインごとにドメイン管理者が定義し，ラベルはドメインに参加するラベル対象 AC とリソースに添付される．

ドメイン管理者はドメインを作成する（ラベルを定義する）と，ドメインで管理したいリソースにラベルを付与する．ドメイン対象外 AC はラベルが付与されたリソースにアクセスすることができない．ラベル対象 AC がドメインに参加する（ドメインのラベルがラベル対象 AC に付与される）と，以後，ラベル対象 AC は同一ラベルが付与されたリソースをアクセスすることが許可される．ラベル対象 AC が利益相反の関係にあるラベルが付与されたリソースにアクセスするためには，ラベル付与解除を行うために，別のアプリケーションインスタンスの起動，つまり現在のラベル対象 AC を終了させて新たなラベル対象 AC を起動させることが必要となる．ドメイン管理者がドメインを削除する（ラベルを無効にする）と，対象ドメイン内のすべてのラベル対象 AC とリソースはラベル付与解除を行う．

プログラムリファレンスモニタを利用して既存環境への影響を最小限に抑えてプロセス隔離を行ううえで，プログラムリファレンスモニタが観測できない情報フローを安全に取り扱うことが重要となる．CWPC では，アプリケーショングループを定義して，グループ内のすべてのメンバに対して Chinese-Wall ポリシを適応するという方法でこの問題を解決している（3.4 節参照）．また，利便性の観点からドメインが削除される前にラベル付与解除を行いたいリソースが存在する．CWPC では，対象リソースをアクセスしたことがあるすべてのラベル対象 AC が終了したことを確認できれば，リソースのラベル付与解除を行えるようにしている（3.4 節参照）．

3. 実　　装

OS プロセスに対して強制アクセス制御を行うことができるプラットフォームとして，Windows ベースのプラットフォーム[14),16)]，Linux ベースのプラットフォーム[9)–12)]，FreeBSD ベースのプラットフォーム[13)] がある．本論文では，検証対象として一般に普及している Windows ベースのプラットフォームを選定し，さらに Microsoft Windows 上で頻繁に利用されるオフィスアプリケーションを選定した．そして，Microsoft Windows 上のオフィスアプリケーションによる安全なドキュメント取扱いを実現するプロトタイプシステムを実装した．

本論文では，ユーザが競合関係のドメインに属する文書間で不用意に情報を漏洩することを防ぐことを仮定しており，ユーザが悪意を持って意図的に漏洩することは想定していない．

3.1 情報フロー制御

ドキュメントアクセス制御の利用シナリオの場合，取り扱う情報フロー[8)] はファイル間の直接的な情報フローと，クリップボードを介したファイル間の間接的な情報フローの 2 種類である．ファイルに添付されているラベルは基本的に変化しない．しかし，クリップボードは複数のプロセス間で共有されるノード上に 1 つしか存在しないリソースであるため，クリップボードに添付されているラベルはラベル対象 AC の活動状況に応じて動的に変化する．そのため，ラベル対象 AC の活動状況に応じてアクセス制御方法を変える必要がある間接的な情報フローを特に注意して取り扱う必要がある．

3.2 Windows 上のアーキテクチャ

プロトタイプシステムのアーキテクチャを**図 3** に示す．Chinese-Wall ポリシのサブジェクトであるラベル対象 AC はオフィスアプリケーションのインスタンスである．オフィスアプリケーションは SI が保持するオフィスアプリケーションリストに事前に定義されている．

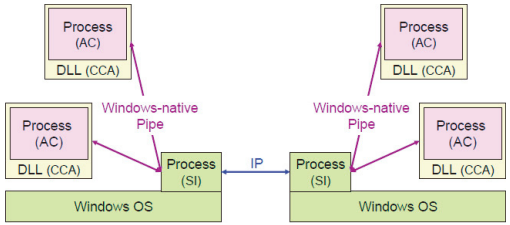


図 3　プロトタイプシステム
Fig. 3　Prototye system.

オフィスアプリケーションの実例として，Microsoft Word，Microsoft Excel，Microsoft PowerPoint，Adobe Acrobat があげられる．ラベル対象外 AC はラベル対象 AC，SI 以外の Windows プロセスである．Chinese-Wall ポリシのオブジェクトであるリソースはディスク上のファイルとメモリ上のクリップボードである．Windows プロセスに添付する CCA はダイナミック・リンク・ライブラリ（Dynamic Link Library，DLL）として実装した．CCA はリソースにアクセスに関する API をフックして，ドメインに特化したアクセス制御を適用する．各 CCA は通信オーバヘッドが小さい Windows-native Pipe で SI と通信する．他の通信は IP チャネルを使って行われる．リソースグループリストは SI 間で共有される．リソースグループリストは SI 間で分散管理，もしくは単一サーバ上で集中管理される．

3.3　リソースへの強制アクセス制御

Windows 環境下ではカーネル空間アプローチはユーザオブジェクトによるクリップボードアクセスを制御できないため，本論文ではユーザ空間アプローチを採用した．

API フック技術としていくつかの研究が行われているが[17]，本論文では OS やアプリケーションのバイナリイメージを変更することなくアプリケーションを制御できる Detours[18] を利用した．Detours はメモリ上の関数イメージを書き換えることによって任意の Win32 関数の制御を横取りするライブラリである．そのため，リソースアクセスを行う Win32 サブシステムモジュールに対して Detours を適応すると，リソースアクセスをフックすることが可能となる．

すべての AC のファイルやクリップボードに対する強制アクセス制御を行うために，すべての AC に対して CCA を添付する必要がある．本論文では，Detours を使ってすべての Windows プロセスに対して強制アクセス制御を行う最新研究[14] を利用した．手順は次のとおりである．CCA はプロセス生成 API である，`kernel32.dll` が提供する CreateProcess 関数と `advapi32.dll` が提供する CreateProcessAsUser 関数を監視する．そして，CCA はプロセスが子プロセスを生成するたびに再帰的に添付する．本論文では，Windows 初期化時に最初に起動する `winlogon.exe` プロセスに CCA を添付させる．その結果，すべてのプロセスに CCA を添付させることができる．各 CCA はプロセスに添付した後，ファイルアクセスに関する Win32 サブシステムモジュール `kernel32.dll` と OS レベルクリップボードアクセスに関する Win32 サブシステムモジュール `user32.dll` を書き換え，ファイルアクセスに関する API（CreateFile，CloseHandle，CopyFile，CopyFileEx，MoveFile，MoveFileEx，MoveFileWithProgress，DeleteFile）と，クリップボードアクセスに関する API（OpenClipboard，CloseClipboard，SetClipboardData，GetClipboardData）を制御する．

3.4　プロセス隔離制御

AC が起動すると CCA が自動的に AC に添付される．CCA は，実行ファイル名，ユーザアカウント名，プロセス ID といった AC に関する情報を SI に送付する．SI は，オフィスアプリケーションリストを用いて AC がラベル対象 AC かラベル対象外 AC かを判別して，ノード上のすべての AC に関する情報を管理する AC 管理リストを更新する．AC がファイルや OS レベルクリップボードにアクセスするために API を呼び出したら，AC に添付している CCA は AC の制御を横取りして，SI にリソースがドメイン管理下のものかどうかを尋ねる．

ファイルに対するアクセス判定アルゴリズムを**図 4** に示す．ファイルがドメインに管理されている場合，ラベル対象 AC はまず，ファイルと異なる利益相反グループのドメインに参加しているときは，ファイルへのアクセスが書き込みならアクセスが拒否され（1–5 行目），読み込みなら許可される（6–7 行目）．そして，ファイルと同一の利益相反グループのドメインに参加していないときは，アクセスが許可され，ファイルと同一のドメインに参加する（8–10 行目）．さらに，ファイルと同一の利益相反グループのドメインに参加しているときはアクセスが許可される（11–12 行目）．それ以外はアクセスが拒否される（13–14 行目）．ラベル対象外 AC はアクセスが拒否される（15–16 行目）．ファイルがドメイン管理下ではない場合，AC がラベル対象 AC で，かつドメインに参加していて書き込みアクセスのときはアクセスが拒否され（17–19 行目），それ以外は許可される（20–21 行目）．

OS レベルクリップボードに対するアクセス判定アルゴリズムを**図 5** に示す．クリップボードにデータがない場合，ラベル対象 AC は複数のドメインに参加していて書き込みアクセスのときはアクセスが拒否され（1–4 行目），それ以外はアクセスが許可される（5–6

```
01  if (File is under domain management)
02     if (AC is a label-attachable AC)
03        if (AC has joined domains in other coalition-of-interest class as File)
04           if (AC's access is write)
05              CCA rejects access;
06           else if (AC's access is read)
07              CCA permits access;
08        else if (AC hasn't joined a domain in the same coalition-of-interest class
                                                                              as File)
09           CCA permits access;
10           CCA makes the AC join the same domain as File and permits access;
11        else if (AC has joined the same domain as File)
12           CCA permits access;
13        else
14           CCA rejects access;
15     else if (AC is a label-free AC)
16        CCA rejects access;
17  else if (File is not under domain management)
18     if (AC is a label-attachable AC, has joined domains, and its access is write)
19        CCA rejects access;
20     else
21        CCA permits access;
```

図 4　ファイルに対するアクセス判定アルゴリズム
Fig. 4　Access decision algorithm for file.

```
01  if (data is not contained in Clipboard)
02     if (AC is a label-attachable AC)
03        if (AC has joined more than two domains and its access is write)
04           CCA rejects access;
05        else
06           CCA permits access;
07           if (AC's access is write)
08              SI makes Clipboard join the same domain as the AC;
09     else if (AC is a label-free AC)
10        CCA permits access;
11  else if (data under domain management is contained in Clipboard)
12     if (AC is a label-attachable AC)
13        if (AC has joined differnt domains from the data)
14           CCA rejects access
15        else if (AC hasn't joined a domain in the same coalition-of-interest class
                                                                        as the data)
16           CCA permits access;
17           CCA makes the AC join the same domain as the data;
18        else if (AC has joined the same domain as the data)
19           CCA permits access;
20        else
21           CCA rejects access;
22     else if (AC is a label-free AC)
23        CCA rejects access;
24  else if (data not under domain management is contained in Clipboard)
25     if (AC is a label-attachable AC and has joined domains)
26        CCA rejects access;
27     else
28        CCA permits access;
```

図 5　OS レベルクリップボードに対するアクセス判定アルゴリズム
Fig. 5　Access decision algorithm for OS-level clipboard.

行目）．その際，ラベル対象 AC がドメインに参加していて書き込みアクセスのときは，クリップボードはラベル対象 AC と同じドメインに参加する（7–8 行目）．ラベル対象外 AC はアクセスが許可される（9–10 行目）．クリップボードにドメイン管理下のデータが含まれている場合，ラベル対象 AC がデータと異なるドメインに参加しているときはアクセスが拒否される（11–14 行目）．そして，データと同一の利益相反グループのドメインに参加しているときはアクセスが許可され，データと同一のドメインに参加する（15–17 行目）．さらに，ファイルと同一のドメインに参加しているときはアクセスが許可される（18–19 行目）．それ以外はアクセスが拒否される（20–21 行目）．ラベル対象外 AC はアクセスが拒否される（22–23 行目）．クリップボードにドメイン管理外のデータが含まれている場合，AC がラベル対象 AC でドメインに参加しているときはアクセスが拒否され（24–26 行目），それ以外は許可される（27–28 行目）．

ドキュメントアクセス制御の利用シナリオにおいて Chinese-Wall ポリシをグループに適応してプロセス隔離を取り扱う必要があるリソースは，アプリケーションレベルクリップボードである．オフィスアプリケーションの中には，アプリケーションレベルのクリップボードを同一アプリケーショングループ内で共有するものがある．この場合，アプリケーションレベルクリップボードへのデータコピー操作は OS レベルクリップボードが利用されるためにプログラムリファレンスモニタで検出できるが，アプリケーションレベルクリップボードからのデータペースト操作は OS レベルクリップボードが利用されないため検出できない．そのため，アプリケーショングループ内のラベル対象 AC がドメインに参加した場合，同一グループのすべてのラベル対象 AC を同一ドメインに参加させるようにした．つまり Chinese-Wall ポリシのラベルが同一アプリケーショングループに属するすべてのサブジェクトに対して添付される．たとえば，Microsoft PowerPoint のインスタンスが $Domain_1$ に参加すると，Microsoft Word や Microsoft Excel といった Microsoft Office グループに属するすべてのアプリケーションのインスタンスが $Domain_1$ に参加することになる．アプリケーションレベルクリップボードを所有するアプリケーションは Microsoft Office や OpenOffice.org など少数であるため，事前にリストを作ることができる．

同一ノード上のプロセス間で共有される OS レベルクリップボードはドメインが削除される前にラベル付与解除を行いたいリソースである．SI はクリップボードがドメインに参加している間，クリップボードにアクセスしたことがあるラベル対象 AC が動作しているかどうかを調べて，すべてのラベル対象 AC の終了が確認できれば，クリップボードのラベル付与解除を行う．

3.5　シナリオ例

シナリオ例を**図 6** に示す．利益相反の関係である $Company_{green}$ と $Company_{red}$ に関するドキュメントが $FileServer$ 上に保管されている．$Company_{green}$ に関するドキュメン

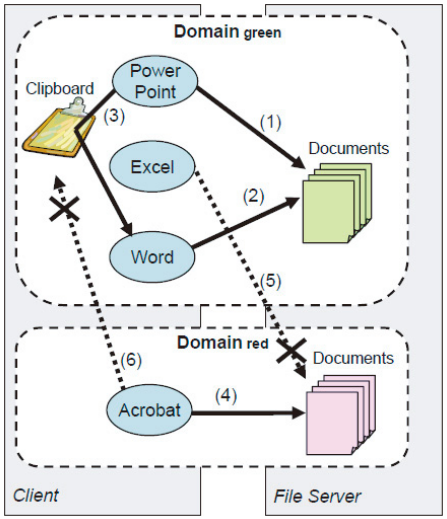


図 6　シナリオ例
Fig. 6　An example of a scenario.

トは $Domain_{green}$ で，$Company_{red}$ に関するドキュメントは $Domain_{red}$ で管理されている．*Client* が *FileServer* 上のドキュメントへのアクセスを試みる．

(1)　*Client* は $Company_{green}$ に関する Microsoft PowerPoint ドキュメントを開こうとする．すると，Microsoft Office グループ内のすべてのアプリケーションは $Domain_{green}$ に参加して，ファイルのオープンに成功する．

(2)　*Client* は $Company_{green}$ に関する Microsoft Word ドキュメントを開こうとする．Microsoft Word はすでに $Domain_{green}$ に参加しているため，ファイルのオープンに成功する．

(3)　*Client* は $Company_{green}$ に関する Microsoft PowerPoint ドキュメント内の図を $Company_{green}$ に関する Microsoft Word ドキュメントにコピー&ペーストしようとする．すると OS レベルクリップボードが $Domain_{green}$ に参加して，コピー&ペースト操作は成功する．

(4)　*Client* は $Company_{red}$ に関する Portable Document Format（PDF）ドキュメントを開こうとする．すると，Adobe Acrobat が $Domain_{red}$ に参加して，ファイルのオープンに成功する．

```
01    <ChineseWallPolicyDefinition>
02       <ChineseWallLabels>
03          <Label>green</Label>
04          <Label>red</Label>
05       </ChineseWallLabels>
06       <ChineseWallSets>
07          <ResourceConflictSet name="Competition1">
08             <Label>green</Label>
09             <Label>red</Label>
10          </ResourceConflictSet>
11       </ChineseWallSets>
12    </ChineseWallPolicyDefinition>
13    ·····
14    <ApplicationTypeDefinition>
15       <ApplicationType name="MicrosoftWord" pattern="WINWORD.EXE"
                                                          isLabelAttachable="true"/>
16       <ApplicationType name="MicrosoftExcel" pattern="EXCEL.EXE"
                                                          isLabelAttachable="true"/>
17       <ApplicationType name="MicrosoftPowerPoint" pattern="POWERPPT.EXE"
                                                          isLabelAttachable="true"/>
18       <ApplicationType name="AdobeAcrobat" pattern="ACROBAT.EXE"
                                                          isLabelAttachable="true"/>
19       ·····
20    </ApplicationTypeDefinition>
21    ·····
22    <ApplicationGroupDefinition>
23       <ApplicationGroupSet name="MicrosoftOffice">
24          <ApplicationType type="MicrosoftWord"/>
25          <ApplicationType type="MicrosoftExcel"/>
26          <ApplicationType type="MicrosoftPowerPoint"/>
27       </ApplicationGroupSet>
28       ·····
29    </ApplicationGroupDefinition>
```

図 7　オフィスアプリケーションシナリオにおけるポリシの例
Fig. 7　An example of the policy for Office application scenario.

(5)　*Client* は $Company_{red}$ に関する Microsoft Excel ドキュメントを開こうとする．しかし，Microsoft Excel はすでに $Domain_{green}$ に参加しているため，ファイルのオープンに失敗する．

(6)　*Client* は $Company_{red}$ に関する PDF ドキュメントから数行の文章をコピーしようとする．しかし，OS レベルクリップボードがすでに $Domain_{green}$ に参加しているため，コピー操作は失敗する．

(7)　*Client* はすべての Microsoft Office アプリケーションのインスタンスを終了する．すると，OS レベルクリップボードは $Domain_{green}$ から離脱する．そして，$Company_{red}$ に関する PDF ドキュメントから数行の文章をコピーする操作は，OS レベルクリップボードが $Domain_{red}$ に参加した後，成功する．

オフィスアプリケーションシナリオにおけるポリシの例を**図 7** に示す．ポリシは，ラベルと利益相反の定義部分（1–12 行目），アプリケーションタイプの定義（14–20 行目），アプリケーショングループの定義（22–29 行目）の 3 つのパートから構成される．最初のパートでは，

green と red という 2 つのラベルと Competition1 という名前の利益相反が宣言されている，次のパートでは，アプリケーションタイプが宣言されている．ApplicationTypeDefinition が宣言されていないアプリケーションのインスタンスはラベル対象外 AC と見なされる．最後のパートでは，MicrosoftOffice という名前のアプリケーショングループが宣言されている．

図 7 より，顧客と一緒に仕事をしているビジネス担当者がラベルと利益相反を定義を行い，システム構成の詳細を知っている IT 管理者がアプリケーションタイプとアプリケーショングループを定義を行うといった，ポリシ管理の分離が可能であることが分かる．

4. 評　　価

ベンチマークプログラムを利用してプロトタイプシステムのオーバヘッドを測定した．測定に使用したマシーンは Lenovo ThinkPad T60 2007-6EJ モデルである．このマシーンはプロセッサに Intel Core Duo T2500 2 GHz を，メモリに 1.5 GByte の PC2-5300 DDR2 SDRAM を搭載していて，Microsoft Windows XP Service Pack2 が動作している．ベンチマークプログラムとして，商用ビジネスアプリケーションを使って性能を評価する際に最もよく使われているベンチマークプログラムの 1 つである Ziff Davis Media Winbench プログラム[19] を利用した．

CWPC 機能が無効の状態（オリジナルの Windows）と有効の状態について測定して比較した（**図 8**）．その結果，グラフィックスアクセスのオーバヘッドはなかったが，ディスクアクセスのオーバヘッドは Winbench Business Disk テストケースの場合で 16.3%，Winbench High-End Disk テストケースの場合で 18.1%という結果になった．Winbench プログラムでは典型的な日常のビジネス用途ではほとんど発生しない高負荷のディスクアクセスを発生させて測定しているため，このオーバヘッドは許容範囲だと考えられる．

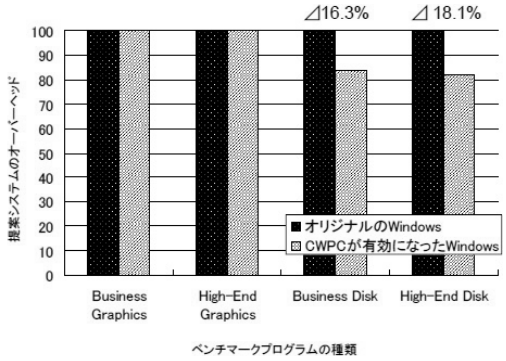


図 8　オーバヘッドの測定
Fig. 8　Overhead evaluation.

5. 関 連 研 究

分散セキュアグループのプロトタイプとして，Terra[1]，NetTop[2]，Shamon[3]，European Multilaterally Secure Computing Base（EMSCB）[4] などが提案されている．Terra[1] は，信頼できる仮想マシンモニタ上で複数の仮想マシンを隔離実行し，仮想マシンのバイナリイメージを使用した構成検証を行う．NetTop[2] は，実行環境を隔離するために VMWare を使用し，VMWare のホスト OS を SELinux といったセキュア OS を利用することでゲスト OS が使用するシステムリソースへのアクセス制御を行っている．Shamon[3] は，複数のノード上で強制アクセス制御ポリシを適応する分散高信頼リファレンスモニタを利用した分散セキュアグループを提案し，プロトタイプシステムを強制アクセス制御を提供する sHype[20] を有効にした Xen[21] の上で実装している．European Multilaterally Secure Computing Base（EMSCB）[4] は，トラステッド・コンピューティング・グループ技術[22] に基づく高信頼コンピューティングとマイクロカーネルに基づく仮想化を組み合わせることによって，多面的セキュリティを実現している．これらにはドメインに参加する仮想マシン，つまり AC を本論文のように隔離ポリシに基づくグルーピングによって隔離する概念がない．そのため本論文で提案した CWPC をこれらプロトタイプに適応することによって，仮想マシンの隔離をさらに強固にすることができると考えられる．

本論文では，OS プロセスのリソースに対する強制アクセス制御の実現に API をフックする方法を用いているが，その他の方法として Fraser らが提案するプロセス・ラッパ[15] がある．プロセス・ラッパは，通常の OS にプロセスのラッパを導入することによってプロセスへの影響を最小限にした強制アクセス制御を実現し，Solaris と FreeBSD 上でプロトタイプシステムが実装されている．

隔離実行環境を実現する研究がいくつか行われている[23)–26)]．SoftwarePot[27),28)] は，インターネットのような安全ではないネットワーク上で流通するソフトウェアを安全に実行するためにソフトウェアを仮想的な環境で隔離実行する機構を提供している．HyperSpector[29] は，分散侵入検知システムを他のシステムから分離して運用するために仮想マシンモニタによってシステムを安全に隔離する仮想分散環境を提案している．これらの研究成果を活用す

ることによって，さらに強固な隔離を実現した分散セキュアグループを構築することができると考えられる．

Chinese-Wall ポリシを分散環境に適応する研究がいくつか行われている．Minsky[30] は，Law Governened Interaction（LGI）機構[31] に対して Chinese-Wall ポリシを規模拡張性高く強制適応する方法を提案している．Atluri ら[32] は分散化されたワークフローに対して Chinese-Wall ポリシを適応する方法を提案している．しかし，両方とも具体的な実装方法について述べられていない．

6. おわりに

本論文では，OS プロセスに対するリソースへの強制アクセス制御技術と連動して，既存 OS やアプリケーションのコードを変更しないでアプリケーションレベル分散セキュアグループを構築するための Chinese-Wall ポリシモデルに基づくプロセス隔離手法 CWPC（Chinese-Wall Policy Confinement）を提案した．さらに，広く普及する Microsoft Windows 環境，およびその上で頻繁に利用されるオフィスアプリケーションの上で安全なドキュメント取扱いを実現するプロトタイプシステムを実装し評価することで，既存環境に対する変更を最小限に抑えられることを示した．

本論文のプロトタイプシステムはファイルとクリップボードへの強制アクセス機能を実装して，ドキュメントの安全な取扱いを実現した．今後，ソケットといった他のリソースに対する強制アクセス機能を実装すれば，他の利用ケース，特にグリッドコンピューティングに適用することができて有用である．他のリソースへの対応はファイルやクリップボードと同様のリソースアクセスに関する API フックによって実現されると考えられる．また，Microsoft Windows 以外の他のプラットフォーム，特に Linux に対応することも有用である．Linux サポートは Linux Security Module（LSM）[33] のフック技術[10)–12)] を使って実現されると考えられる．最後に，本論文では提案手法がすべてのノードで正しく動作していることを前提としているが，提案手法が正しく動作していることの保証・検証を行う機構を導入すれば，さらに実用的な分散セキュアグループが構築できると考えられる．

謝辞　本研究は，経済産業省，新世代情報セキュリティ研究開発事業の研究として行われたものである．

参 考 文 献

1) Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M. and Boneh, D.: Terra: A Virtual Machine-based Platform for Trusted Computing, *Proc. 19th Symposium on Operating System Principles* (*SOSP 2003*) (2003).
2) Meushaw, R. and Simard, D.: NetTop: Commercial Technology in High Assurance Applications, *Tech. Trend Notes Volume 9 Edition 4*, National Security Agency (2000).
3) McCune, J.M., Berger, S., Caceres, R., Jaeger, T. and Sailer, R.: Shamon: A System for Distributed Mandatory Access Control, *Proc. 22nd Annual Computer Security Applications Conference* (*ACSAC 2006*) (2006).
4) Sadeghi, A.-R. and Stuble, C.: Towards Multilateral Security on DRM Platforms, *Proc. 1st Information Security Practice and Experience Conference* (*ISPEC 2005*) (2005).
5) Lampson, B.W.: A Note on the Confinement Problem, *Comm. ACM*, Vol.16, No.10 (1973).
6) Brener, D.F.C. and Nash, M.J.: The Chinese Wall Security Policy, *Proc. Symposium on Research in Security and Privacy*, Oakland (1989).
7) Bell, D.E. and Padula, L.J.L.: Secure Computer System: Unified Exposition and Multics Interpretation, MITRE Report MTR 2547 (Nov. 1973).
8) Myers, A.C. and Liskov, B.: A Decentralized Model for Information Flow Control, *Proc. 16th ACM Symposium on Operating Systems Principles* (*SOSP '97*) (1997).
9) Loscocco, P. and Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System, *FREENIX Track: 2001 USENIX Annual Technical Conference* (2001).
10) Jaeger, T., Edwards, A. and Zhang, X.: Consistency Analysis of Authorization Hook Placement in the Linux Security Modules Framework, *ACM Trans. Information and System Security* (*TISSEC*), Vol.7, Issue 2 (2004).
11) Zhang, X., Edwards, A. and Jaeger, T.: Using CQUAL for Static Analysis of Authorization Hook Placement, *Proc. 11th USENIX Security Symposium* (2002).
12) Edwards, A., Jaeger, T. and Zhang, X.: Runtime Verification of Authorization Hook Placement for the Linux Security Modules Framework, *Proc. 9th ACM Conference on Computer and Communications Security* (*CCS 2002*) (2002).
13) Watson, R., Morrison, W., Vance, C. and Feldman, B.: The TrustedBSD MAC Framework: Extensible Kernel Access Control for FreeBSD 5.0, *Proc. USENIX 2003 Annual Technical Conference* (2003).
14) 古市実裕，三品拓也，大谷　佑：PC におけるプログラムのリアルタイム監視・制御技術の開発と情報セキュリティシステムへの応用，暗号と情報セキュリティシンポジウ

ム（SCIS2006）予稿集 (2006).
15) Fraser, T., Badger, L. and Feldman, M.: Hardening COTS Software with Generic Software Wrappers, *Proc. IEEE Symposium on Security and Privacy*, Oakland (1999).
16) 神田勝規，大山恵弘，光山義紀，加藤和彦：Windows におけるリファレンスモニタの実現，情報処理学会研究報告：システムソフトウェアとオペレーティング・システム (2003).
17) 安藤類央，外山英夫，門林雄基：DLL Injection による P2P ソフトウェアの情報漏洩の追跡と防止，情報処理学会コンピュータセキュリティ研究会第 31 回研究報告 (2007).
18) Hunt, G. and Brubacher, D.: Detours: Binary Interception of Win32 Functions, *Proc. 3rd USENIX Windows NT Symposium* (1999).
19) Ziff Davis Media Winbench. http://www.veritest.com/benchmarks/winbench
20) Sailer, R., Jaeger, T., Valdez, E., Caceres, R., Perez, R., Berger, S., Griffin, J. and van Doorn, L.: Building a MAC-based Security Architecture for the Xen Opensource Hypervisor, *Proc. Annual Computer Security Applications Conference* (*ACSAC 2005*) (2005).
21) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauery, R., Pratt, I. and Warfield, A.: Xen and the Art of Virtualization, *Proc. 19th ACM Symposium on Operating Systems Principles* (*SOSP '03*) (2003).
22) Trusted Computing Group (TCG). http://www.trustedcomputinggroup.org/
23) 尾上浩一，大山恵弘，米澤明憲：セキュリティシステム保護のためのサンドボックスシステム，日本ソフトウェア科学会第 22 回大会 (2005).
24) Garfinkel, T. and Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, *Proc. Symposium on Network and Distributed System Security* (*NDSS 2003*) (2003).
25) Liang, Z., Venkatakrishnan, V.N. and Sekar, R.: Isolated Program Execution: An Application Transparent Approach for Executing Untrusted Programs, *Proc. Annual Computer Security Applications Conference* (*ACSAC 2003*) (2003).
26) Quynh, N.A., Ando, R. and Takefuji, Y.: Centralized Security Policy Support for Virtual Machine, *Proc. 20th USENIX Large Instllation System Administration Conference* (*LISA '06*) (2006).
27) 大山恵弘，神田勝規，加藤和彦：安全なソフトウェア実行システム SoftwarePot の設計と実装，日本ソフトウェア科学会コンピュータソフトウェア，Vol.16, No.6 (2002).
28) Kato, K. and Oyama, Y.: SoftwarePot: An Encapsulated Transferable File System for Secure Software Circulation, *Proc. International Symposium on Software Security* (2003).
29) Kourai, K. and Chiba, S.: HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection, *Proc. 1st ACM/USENIX International Conference on Virtual Execution Environments* (*VEE '05*) (2005).
30) Minsky, N.: A Decentralized Treatment of a Highly Distributed Chinese-Wall Policy, *Proc. 5th IEEE International Workshop on Policies for Distributed Systems and Networks* (2004).
31) Minsky, N. and Ungureanu, V.: Law-Governed Interaction: A Coordination & Control Mechanism for Heterogeneous Distributed Systems, *ACM Trans. Software Engineering and Methodology* (*TOSEM*), Vol.9, Issue 3 (2000).
32) Atluri, V., Chun, S.A. and Mazzoleni, P.: A Chinese-Wall Security Model for Decentralized Workflow Systems, *Proc. 8th ACM Conference on Computer and Communications Security* (*CCS 2001*) (2001).
33) Wright, C., Cowan, C., Morris, J., Smalley, S. and Kroah-Hartman, G.: Linux Security Modules: General Security Support for the Linux Kernel, *Proc. 11th USENIX Security Symposium* (2002).

付　　録

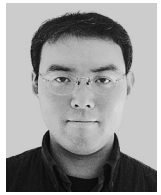
Microsoft，Windows，Windows XP，Word，Excel，PowerPoint は Microsoft Corporation の米国およびその他の国における商標．Adobe，Acrobat は Adobe Systems Incorporated の米国およびその他の国における商標．Linux は Linus Torvalds の米国およびその他の国における商標．他の会社名，製品名およびサービス名などはそれぞれ各社の商標．

(平成 19 年 10 月 5 日受付)
(平成 20 年 4 月 8 日採録)

推　薦　文

本論文は 1 台の PC 内で，誤ってポリシの異なるデータをアプリケーション間でやりとりしない方式に関する論文である．複数のユーザが複数の PC でデータにアクセスする際に，ポリシに従って排他制御する例などへの適用が考えられ，様々な拡張が期待できる．設計・実装主体の実践的・有用性が高い論文であり，かつ，実験による性能評価と関連研究との比較も丁寧に論じられている．論文としての完成度も高く，推薦に値する．

（マルチメディア通信と分散処理研究会主査　櫻井紀彦）



勝野　恭治（正会員）

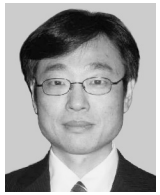
1998 年慶應義塾大学大学院理工学研究科計算機科学専攻修士課程修了．同年日本アイ・ビー・エム株式会社入社．東京基礎研究所主任研究員．2008 年筑波大学大学院システム情報工学研究科リスク工学専攻後期博士課程入学．2003 年ソフトウェア科学会高橋奨励賞受賞．情報セキュリティ，コンピューター・ネットワーク，エージェント技術に関する研究開発に従事．日本ソフトウェア科学会会員．

渡邊　裕治（正会員）

2001 年東京大学大学院工学系研究科電子情報工学専攻博士課程修了．同年日本アイ・ビー・エム株式会社入社．東京基礎研究所主任研究員．東京工業大学非常勤講師．2004 年度情報処理学会論文賞受賞．内部統制，コンプライアンス技術，セキュリティ，プライバシ保護，トレーサビリティに関する研究開発に従事．博士（工学）．

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・エム株式会社入社．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 月に国際標準となる．2002 年東京大学より博士（工学）の学位授与．電子情報通信学会会員．