Vol.2014-CSEC-67 No.4
2014/12/5

セキュリティラベルの文書間伝播と動的セキュリティポリシー切り替えを用いたデスクトップクラウド向け情報漏洩防止システム

三品 拓也[1,a)]　浦本 直彦[1,b)]

概要：デスクトップクラウドは，クラウド上で稼働する仮想デスクトップ上でデータの保管と処理を行い，手元の PC に画面だけを転送するサービスである．その目的のひとつはセキュリティの向上であり，PC に機密度の高い情報が保管されなくなることで，PC 紛失時の被害を回避することができる．しかし，デスクトップクラウドの利用によっても，ユーザが組織のセキュリティポリシーに違反するような操作を無意識もしくは意図的に行うことによって，エンドポイント PC を通過するデータが不適切な場所へ流れていく形での情報漏洩は起こり得る．そこで本研究では，デスクトップクラウドで取り扱われるオフィス文書に対するユーザの操作を，当該文書の機密度に応じて制御することで情報漏洩を防ぐ DLP (Data Lakage Prevension) システムを設計・実装した．このシステムは，文書の内容に応じたセキュリティラベルを自動的に付与する技術と，オペレーティングシステムに対するアプリケーションの API 呼び出しをフックすることでアプリケーションに許可する動作を制御する技術からなる．セキュリティラベルは，文字列パターンを用いて特定のキーワードを抽出することで決定されるほかに，文字列と図形の類似性に基づいて文書間でセキュリティラベルを伝播させることで，うっかりミスを含めた情報漏洩防止を実現する．またアプリケーション制御は，PC の位置やアプリケーションの状態に応じて動的にセキュリティポリシーを切り替えることで，利便性を維持しつつセキュリティを確保できる仕組みとした．

1. はじめに

デスクトップ PC やノート PC といったユーザの手元にある PC（エンドポイント PC）は，大きな価値を持った情報を大量に含んでいる反面，入退室管理がなされていない場所に設置されている・ノート型であらゆる場所に持ち運ぶことができる，といった可搬性の良さゆえに，機器の紛失・盗難によって，機器に搭載された不揮発記憶装置（ハードディスクやソリッドステートドライブ）に記憶されている情報が窃取されるリスクが高い．情報漏洩が発生したときの金銭的・社会的被害は非常に大きいため [*1]，これまでも様々な方法でデータを保護する手段が講じられてきた．もっとも単純な保護手段は安全な場所（例：会社の建物内）からの持ち出し禁止だが，出張や在宅勤務といったオフィス外での業務時に大きな支障を生じるため，様々な技術的解決方法が模索されてきた．例えば全ディスク暗号化は，エンドポイント PC の不揮発記憶装置に記憶されている情報を全て暗号化して使用時に復号する手法であり，電源オフもしくは休止状態で紛失した場合の情報漏洩を防止することができる．この手法には，導入前・導入後でユーザ体験がほとんど変化しないという利点がある一方で，手元にデータがあるという事実に変わりはなく，サスペンド中や画面ロック中に盗難に遭うと，盗難の時点で復号化されていた情報は漏洩し得るし，パスワードが漏洩すると全データが危殆化する欠点が存在する．また，暗号化・復号処理のぶんだけエンドポイント PC のパフォーマンスが低下する．

デスクトップクラウドは，クラウド上で稼働する仮想デスクトップ上でデータの保管と処理を行い，手元の PC に画面だけを転送するサービスである．このサービスを利用する場合，一時的にせよエンドポイント PC に保管されるのは画面表示データのみであり，PC の休止・活動の状態にかかわらず，不揮発記憶装置からのデータ窃取のリスクはほぼゼロにできる [*2]．

[1] 日本アイ・ビー・エム（株）東京基礎研究所
5-6-52 Toyosu, Koto-ku, Tokyo, 135–8511, Japan
[a)] tmishina@jp.ibm.com
[b)] uramoto@jp.ibm.com

[*1] 例えばレポートのひとつ [1] では，138 件の事件から得られた，紛失したラップトップ 1 台の価値は平均で 49246 ドルとされている．

[*2] 画面のキャッシュやエンドポイント PC からデスクトップクラウ

しかし，デスクトップクラウドの利用によっても，ユーザが組織のセキュリティポリシーに違反するような操作を無意識もしくは意図的に行うことによって，エンドポイント PC を通過するデータが不適切な場所へ流れていく形での情報漏洩は起こり得る．

ここで，以下のようなシナリオを想定する．

前提　ある単一の組織が維持・管理しているデスクトップクラウド環境を，その組織の複数ユーザが利用している．その組織はデスクトップクラウド環境とは別に様々な web ベースのアプリケーションを提供しており，ファイル共有のための企業用 SNS，旅費精算システム等が提供されているものとする．この組織では，ファイルのヘッダ部分もしくはフッタ部分に文字列「Confidential」が含まれている場合，そのファイルは機密文書であり，オフィス以外の場所で印刷してはならない，というセキュリティポリシーを持っている，とする．また，成果物の共有・再利用のため，デスクトップクラウド環境からアクセスできるサーバでファイル共有が実施されている．

シナリオ 1　旅費精算を行うため，ユーザは自宅でエンドポイント PC を開き，デスクトップクラウド環境を経由して旅費精算システムにアクセスする．領収書の貼り付け台紙を印刷するように求められたので，旅費精算システムからダウンロードしたファイルを印刷する．

シナリオ 2　ユーザは今年度の投資計画書を自宅のエンドポイント PC で閲覧する．このファイルのヘッダ部分には「Confidential」という文字列が記入されているが，ユーザはこの文字列を見落として印刷を試みる．

シナリオ 3　シナリオ 2 の操作を，自宅ではなくオフィスで行う．

シナリオ 4　ユーザは今年度の投資計画書を開いて内容をコピーし，新しいファイルにペーストして，来年度の投資計画書の原稿として再利用した．しかし，このときいヘッダ部分のコピーを失念し，「Confidential」という文字列がコピーされなかった．このファイルを別のユーザが開き，印刷を試みる．

シナリオ 1 における印刷操作は通常業務の範囲内であり，これが許可されないと業務に支障を生ずる．一方シナリオ 2 では，印刷操作がこの組織のセキュリティポリシーに違反しており，何らかの方法で操作を阻止する必要がある．しかし，ファイル形式やアプリケーションの単位で印刷操作を制御してしまうと，シナリオ 1 における印刷操作まで実行できなくなる．ゆえに，シナリオ 1 とシナリオ 2 とで操作の可否を変えるためには，何らかの形でファイルの内容を解析し，組織のセキュリティポリシーに照らして許可・不許可を判断するしくみが必要である．同様に，シナリオ 3 において印刷操作を阻止するには，エンドポイント PC もしくはプリンタの設置場所を検出し，許可・不許可を判断するしくみが必要である．

シナリオ 4 の例では，図内の「文書 2」を印刷する操作そのものはセキュリティポリシーに違反していない．しかし，「文書 2」の内容は，機密扱いが指示されている「文書 1」の内容そのものを含んでおり，当然機密扱いされるべきである．一般にオフィス文書は複数人によって複数回再利用されるものであり，このような無意識の情報漏洩が起こらないしくみを備えることも望ましい．

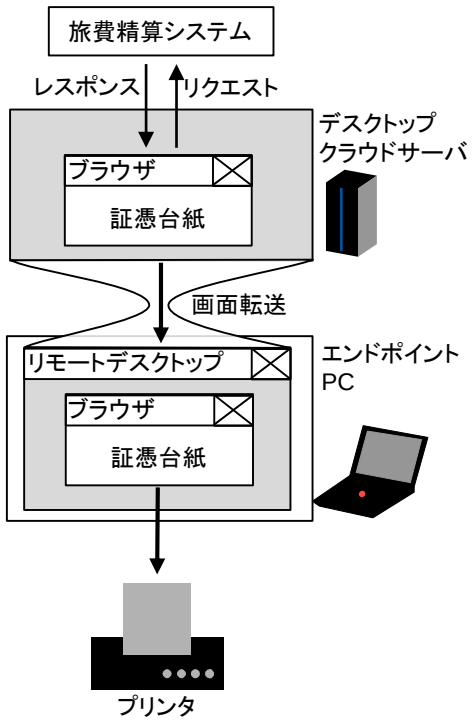


図 1　シナリオ 1

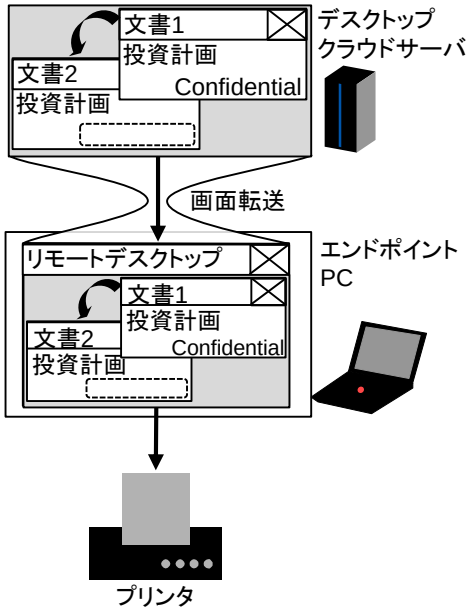


図 2　シナリオ 4

ドサーバへ接続するための設定情報（セキュリティ証明書など）を窃取されるリスクは残る．

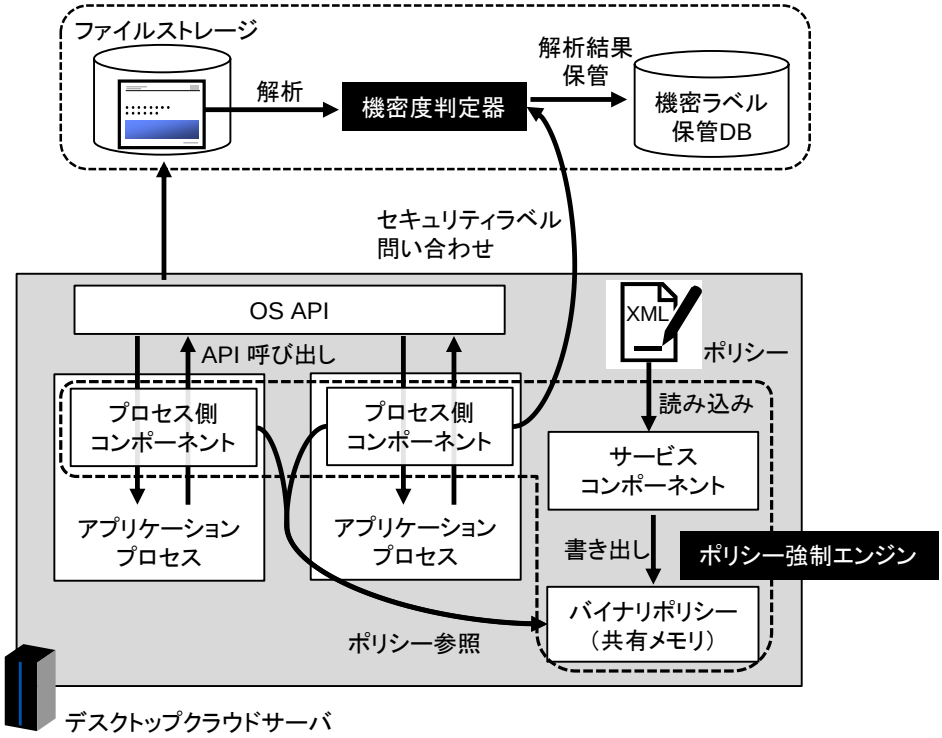


図 3 DLP システムの全体構成

以上のような情報漏洩防止を実現する技術として，情報の機密度をその情報の内容に基づいて判定し，判定の結果得られた機密度に応じた制御を行う DLP (Data Leakage Prevention) をデスクトップクラウドに適用することが考えられる．DLP は，あるデータに対して，ファイルの読み書き・印刷・クリップボードへのコピーなど，計算機上で行われる操作を許可するかどうかを，そのデータの内容から自動的に判定して，情報漏洩など，意図しない情報の流れを抑止する技術・製品の総称である．一般的な DLP の処理は，内容の機密度に即したセキュリティラベルを自動的に決定する機密度解析（Classify），セキュリティラベル・実行コンテキスト・セキュリティポリシーに基づく操作の許可・不許可の判定（Decision），判定結果を強制するためにシステムの振る舞いを変更するポリシー強制（Enforce）の 3 つの手順からなる．デスクトップクラウド環境においては，データはサーバ側で集中管理されているので，この 3 手順のうち機密度解析の実行結果を複数ユーザで共有することができるため，各エンドポイント PC で個別に機密度解析をする場合に比べて少ない計算機資源で DLP を実行することができる．ゆえに DLP はデスクトップクラウドとの親和性が高い技術だと言える．

本研究では，デスクトップクラウドで取り扱われるオフィス文書に対するユーザの操作を，当該文書の機密度に応じて制御することで情報漏洩を防ぐ DLP システムを設計・実装した．このシステムは，文書の内容に応じたセキュリティラベルを自動的に付与する技術と，オペレーティングシステムに対するアプリケーションの API 呼び出しをフックすることでアプリケーションに許可する動作を制御する技術からなる．セキュリティラベルは，文字列パターンを用いて特定のキーワードを抽出することで決定されるほかに，文字列と図形の類似性に基づいて文書間でセキュリティラベルを伝播させることで，うっかりミスを含めた情報漏洩防止を実現する．またアプリケーション制御は，PC の位置やアプリケーションの状態に応じて動的にセキュリティポリシーを切り替えることで，利便性を維持しつつセキュリティを確保できる仕組みとした．

2. 全体構成

今回設計・実装したデスクトップクラウド環境向け DLP システムの全体構成を図 3 に示す．本システムが想定しているデスクトップクラウド環境は，サーバ側・エンドポイント側ともに Microsoft Windows が動作しており，両者を MSRDP (Microsoft Remote Desktop Protocol) で接続して，サーバの画面をエンドポイントへ転送する形である．

本システムは，デスクトップクラウドサーバで動作するポリシー強制エンジンと，それとは別に用意されたサーバで稼働する機密度判定器からなる．ポリシー強制エンジンはデスクトップクラウドサーバで動作するすべてのプロセスの内部に注入され，各プロセスの動作を監視して，必要に応じて機密度判定器をネットワーク越しに呼び出す（4 節参照）．ポリシー強制エンジンは機密度判定によって得られるセキュリティラベルとセキュリティポリシーに基づ

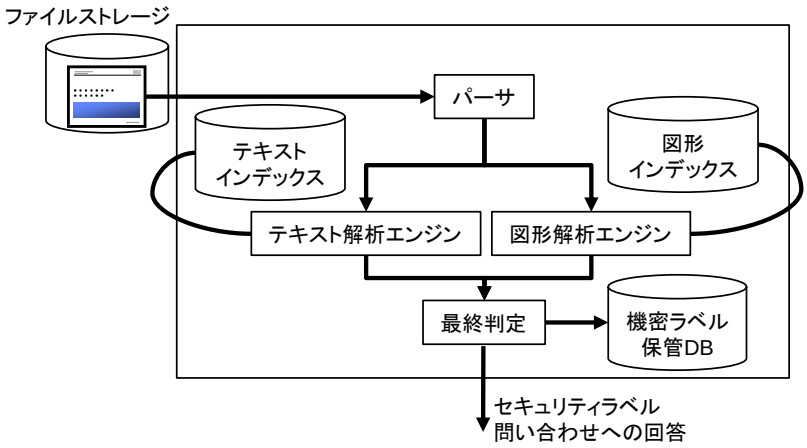


図 4 機密度判定器のコンポーネント群

いて，各プロセスの振る舞いを制御する．

3. 機密度判定器 (Classifier)

機密度判定は，ユーザがファイルに対してなんらかの操作を試みたときに発生する．実際には，特定のアプリケーションがファイルを開こうとしたとき・印刷しようとしたとき・クリップボード操作を行おうとしたときと，エクスプローラでファイルのコピー処理を行おうとしたときをトリガーとしている．

図 4 に，本研究で実装した機密度判定器のコンポーネント群を示す．機密度判定器は，まず対象ファイルが前回機密度判定されたときと同一のデータであるかどうかを確認し，同一である場合は前回の判定結果をキャッシュから取り寄せてそのままポリシー強制エンジンに渡す．異なる場合は，対象ファイルに対して，以下で説明する文字列パターン検出，文字列類似度検出，図形類似度検出を行い，最も機密度の高いセキュリティラベルを対象ファイルのセキュリティラベルであると判定して，ポリシー強制エンジンに渡す．

3.1 セキュリティラベル

本システムでは，ある特定のルールに従った取り扱いをすべきファイルに対して，セキュリティラベルと呼ばれる文字列を付与して識別を行う．どのような内容のファイルにどのセキュリティラベルを付与するのかは，次節にて記述する．

3.2 文字列パターン検出によるセキュリティラベル付与

本システムでは，セキュリティポリシーで定義された文字列パターンが処理対象のファイルに含まれるかどうかを判定する．単純なパターン認識により，「Confidential」といったキーワードの存在を検査する．なお，技術的にはヘッダ・フッタ部分に限定して検索を行うことも可能であるが，予備実験において，本文やテキスト図形としてこのようなキーワードを記入した例が散見されたため，特に検索対象の限定は行わずに検査を行うこととした．

3.3 セキュリティラベルの伝播

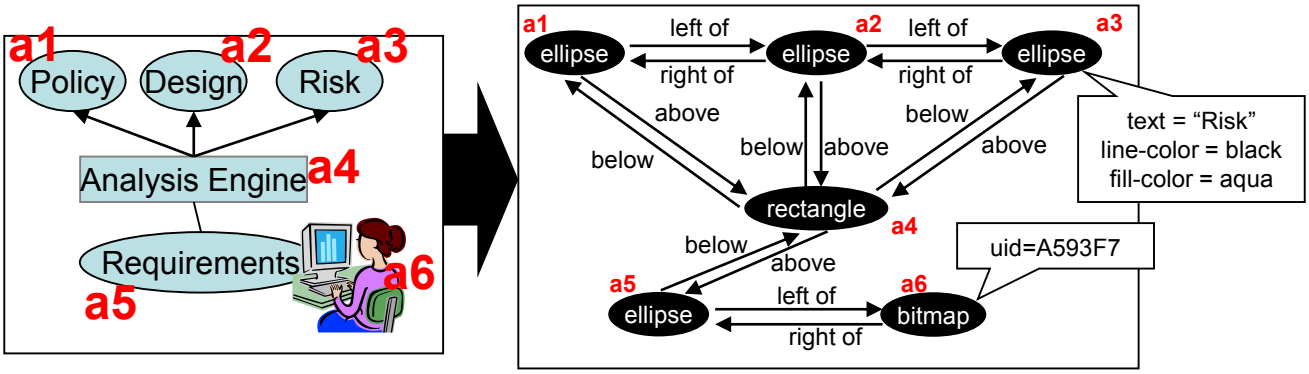
1 節のシナリオ 4 で示した通り，組織のルールとして機密度を示すキーワードをファイル内に記入することが求められている場合であっても，ユーザの不注意によってそれが欠落することがある．このようなキーワードの欠落したファイルに対しても適切なセキュリティラベルを付与するために，本システムでは，あるセキュリティラベルが与えられているファイルと内容が類似している場合は，当該セキュリティラベルを全社のファイルから後者のファイルへ伝播（コピー）させる．このしくみを導入することにより，機密キーワードの記入漏れが起こったファイルに対しても，正しいセキュリティラベルを付与できる確率を高めることができる．どのようにして類似するファイルを検出するのかを，次節以降で記述する．

3.4 ファイル類似度検出 (1) 文字列類似度

本システムでは，全文検索ツール Apache Lucene を利用してファイル間のテキスト類似度を計算する．あらかじめ，ファイル共有が行われているサーバの特定のディレクトリを監視し，そのディレクトリに含まれているファイルが新規登録・内容変更されるたびに，インデックス更新作業を行う．DLP 実行時には，対象ファイルをクエリとしてインデックスを検索し，類似度の指標が一定の閾値を越えたファイルのリストを作成する．検索結果のファイルに対して再帰的に類似度検出を行い，得られたセキュリティラベルのうち，セキュリティポリシーで定義された，最も機密度の高いセキュリティラベルを，対象ファイルのセキュリティラベルと見なして，ポリシー強制エンジンに渡す．

3.5 ファイル類似度検出 (2) オブジェクト類似度

文字列類似度を使った方法は，文字列の多いファイルで



解析対象のページ　　グラフへの変換結果

図 5　プレゼンテーションスライドをグラフ構造に変換する例

は有効に作用すると考えられるが，プレゼンテーションファイルのように，文字よりも絵・イラスト・幾何学図形が文字列より多く含まれているファイルの場合，文字列パターンだけでは適切な類似度検出を行うことが困難であると考え，ファイルに含まれるオブジェクトの類似度を使ったセキュリティラベル伝播も併用することとした．具体的には，あるページに属する全てのオブジェクトについて，ひとつのオブジェクトのひとつのノード，オブジェクトとオブジェクトとの位置関係をエッジとしたグラフ表現に変換して，ランダムウォークを用いたグラフマイニング [2] を適用して，類似するグラフを持つファイルを検索する．この手法も文字列類似度検出と同様に，予め特定のディレクトリに保管されている全てのファイルをグラフ表現に変換しておく必要がある．

図 5 に，プレゼンテーションファイルの 1 ページ（1 スライド）をグラフ構造に変換した例を示す．おおまかな図形配置の類似度を検出するため，オブジェクト同士の関係は，上下左右のいずれかであるかという情報のみを利用する．

ふたつのグラフの類似度指標として使うカーネル関数 $K(G,G')$ は，以下のように定義される．

$$\begin{aligned} K(G,G') = \sum_{\ell=1}^{\infty} \sum_{\boldsymbol{h}} \sum_{\boldsymbol{h'}} & p_s(h_1) \prod_{i=2}^{\ell} p_t(h_i|h_{i-1}) p_q(h_1) \\ & \times p'_s(h'_1) \prod_{j=2}^{\ell} p'_t(h'_j|h'_{j-1}) p'_q(h'_\ell) \\ & \times K(v_{h_1}, v'_{h'_1}) \\ & \prod_{k=2}^{\ell} K(e_{h_{k-1},h_k}, e'_{h'_{k-1},h'_k}) K(v_{h_k}, v'_{h'_k}) \end{aligned} \tag{1}$$

ただし，$p_s(i)$ はランダムウォークをノード i から開始する確率，$p_t(j|i)$ はノード i からノード j への遷移確率，$p_q(i)$ ノード i でランダムウォークが終了する確率，$K(v,v')$ はノードペア $(v,\ v')$ に適用されるカーネル関数，$K(e,e')$ エッジペア (e,e') に適用されるカーネル関数である．$p_s(i)$, $p_t(j|i)$, $p_q(i)$ はランダムウォークの特徴を決める関数であり，確率としての制約（$0 \le p(x) \le 1$ かつ $\sum p(x) = 1$）を満たす任意の関数を使うことができる．今回の実装では以下を用いた．

$$\begin{aligned} p_s(i) &= \frac{\mathrm{area}(i)}{\sum_{v \in G} \mathrm{area}(v)} \\ p_t(j|i) &= \frac{\mathrm{area}(j)}{\sum_{k \in V(i)} \mathrm{area}(k)} \\ p_q(i) &= \mathrm{const.} \end{aligned} \tag{2}$$

ここで $\mathrm{area}(i)$ は，i 番目のノードが指しているオブジェクトの面積を表す．これは，面積が大きいオブジェクトの類似度をより重視するために，面積が大きいオブジェクトに対してより高頻度にランダムウォークを行うべきことを指示している．

$K(e,e')$ と $K(v,v')$ は，それぞれエッジとノードの類似度を表す関数である．カーネル関数としての性質（$K(x,x') = K(x',x)$, $K(x,x') > 0$ など．詳細は文献 [3] 参照）を満たす任意の関数を用いることができる．そこで，$K(v,v')$ は，オブジェクトに付与されたテキストラベル・横幅・縦幅・オブジェクト種別の類似度の相加平均（これらが全て同一であれば 1，全て異なれば 0）を，$K_e(e,e')$ としては，エッジラベルが同一であれば 1，そうでなければ 0 を返す関数を与えた．

4. ポリシー強制エンジン (Enforcement)

4.1 概要

本システムにおけるポリシー強制エンジンは，主に 2 つのコンポーネントからなる．ひとつは各アプリケーションプログラムに強制的に注入されて，アプリケーションプロ

グラムのプロセスに所望の振る舞いを強制させるプロセス側コンポーネントであり，もうひとつはサービスプログラムとしてOSに常駐するポリシー処理を担当するサービスコンポーネントである．

OSが起動すると，まずはサービスコンポーネントがXMLファイルで記述されたポリシーファイルを読み込み，共有メモリ上に展開する．このサービスコンポーネントは，起動するアプリケーションプログラム全てにプロセス側コンポーネントが注入する作業も行う．注入されたプロセス側コンポーネントは，各アプリケーションプログラムがMicrosoft Windowsの各種API (win32.dll, gdi32++.dll, winsock.dll) を呼び出す操作をフックすることで，特定の操作の許可・不許可を制御する．アプリケーション側から発行されるAPI呼び出しを許可する場合は呼び出しをそのままOS側に伝達し，許可しない場合はOS側のAPIを呼び出さずに，ポリシー強制エンジンが自己判断でエラーメッセージをアプリケーションに返す．

4.2 アプリケーションプラグインによるアプリケーションコンテキストの取得

Microsoft WindowsのようなGUI OSでは，複数のファイルを同時に開いて複数のウィンドウにファイルを割り当てることで，複数のファイルを同時に処理できるようになっているアプリケーションが多数存在する．その場合，ある操作を許可するか否かを決めるために使われるべきファイルは，その操作が行われようとしているときにもっとも前面に表示されているウィンドウが関連づけられているファイルである．しかし，どのウィンドウにどのファイルが割り当てられているかはそのアプリケーションしか知り得えないので，図6に示したように，各アプリケーション毎にファイルとウィンドウの関係を記録し，サービスコンポーネントに報告するプラグインが必要になる．現状ではMicrosoft Word, 同Excel, 同PowerPoint及びInternet Explorer向けプラグインが実装済となっている．

なお，本システムで用いているポリシー強制エンジンは，もともとシステム全体の振る舞いを監視する目的で実装されたため，ポリシーの内部表現は共有メモリ上のバイト列（フラグの配列）であり，全てのプロセス側コンポーネントが単一のポリシーを参照している．一方DLPでは，保護対象のファイルごとに異なるポリシーを適用する必要があるので，アプリケーションの切り替えや，ウィンドウの切り替えに応じて，適切なセキュリティポリシーをその都度読み込むように改変を行っている．

4.3 ポリシーモデル

本システムにおけるポリシーは，アクセス制御モデルのオープンスタンダードであるXACML (eXtensible Access Control Markup Language)[4] で規定されている

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="{5E982CBB-xxxx-xxxx-xxxx-550
    D4C6B8425}"
 RuleCombiningAlgId="identifier:rule-combining-
     algorithm:deny-overrides">
 <Description>secret mode policy</Description>
...
 <Rule RuleId="run:ibm:rule1:Default">
   <Description>default</Description>
   <Subjects><AnySubject/></Subjects>
   <Resources><AnyResource/></Resources>
 </Rule>
 <Rule RuleId="run:ibm:rule1:test">
   <Description>notepad</Description>
   <Subjects>
     <Subject>
       <AttributeValue DataType="file:path">notepad
           .exe</AttributeValue>
     </Subject>
   </Subjects>
   <Resources>
   <!------- (A) -------->
     <Resource DataType="file" Lib="SBLFILE.DLL"
         Target="Local">
       <AttributeValue DataType="content:category">
           confidential</AttributeValue>
       <AttributeValue DataType="file:path">\\
           fileserver.example.com\daasdemo\*</
           AttributeValue>
       <Action Effect="Deny">write</Action>
       <Obligation DataType="Popup"/>
     </Resource>
 <!------- (B) -------->
 <Rule RuleId="run:ibm:rule1">
   <Description></Description>
   <Subjects><AnySubject/></Subjects>
   <Obligations>
     <Resources>
       <Resource DataType="print" Lib="SBLPRINT.DLL
           " Target="Local">
         <AttributeValue DataType="printer:address"
             >*.example.com</AttributeValue>
         <Action Effect="Permit">print</Action>
       </Resource>
     </Resources>
   </Obligations>
 </Rule>
</Policy>
```

図 7 セキュリティポリシー例

ポリシーファイルを，Windows実装向けに改変した形式で表現する．操作制限は，システム全体に適用されるルールと，プロセス単位のルールの2種類を記述することができる．プロセス単位のルールでは，当該プロセスが特定のリソース（ファイル，プリンタ，クリップボード，スクリーンショット）を取り扱うときのルールを，リソース毎に記述することができる．図7(A)はその一例である．この指定では，プロセス`notepad.exe`に対して，`confidential`というカテゴリ（セキュリティラベル）が機密度判定器から付与されていて，かつファイルパスが`\\fileserver.example.com\daasdemo\*`に合致する場合，ファイルへの書き込みを拒否するルールを記述している．図7(B)では，印刷に関するポリシーを記述して

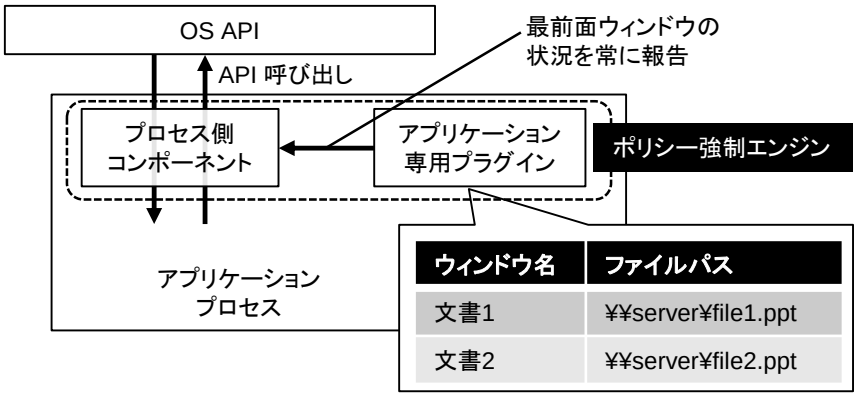


図 6 アプリケーションプラグインからポリシー強制エンジンへのアプリケーション内部状態通知

いる．ここではプリンタの IP アドレスを指定して，印刷を許可するプリンタを指定している．この場合，システム全体に適用されるルールとしては印刷を拒否するルール（“print” の代わりに “deny” ）を指定することで，特定のアドレスのプリンタだけに印刷を許可することができる．

5. 関連研究

本研究は，テキスト検索によるキーワード検出に，テキストと図形の類似度に基づくセキュリティラベルの伝播を組み合わせて，情報漏洩の恐れのある文書をなるべく洩れなく検出し，状況に応じて必要最小限の制限を加えることで，セキュリティとユーザビリティの両立を図っている．テキスト類似度の計算に検索エンジンの技術を用いる先行研究としては田代ら [5] の研究が挙げられる．また，Metzler ら [6] では，本研究で用いた TF-IDF を含む様々な比較手法について，その性能を比較している．特定のファイル形式に特化した類似検索としては，最近のオフィス文書（OpenDocumentFormat[7] や Office Open XML[8]）で採用されている XML に特化した比較手法 [9][10] が提案されている．

図形類似度の検出手法は，ビットマップとして比較する手法とベクトル画像として比較する手法に大別される．前者については Metzler らによるサーベイ論文 [11] が詳しい．本研究ではグラフを使ってベクトル画像を表現し，その全体の類似度を比較したが，Huan らのサブグラフ検索手法 [12] では，全体ではなく部分の類似度を検索することができるので，特定の図形集合を特に検索したい場合はこの手法の適用も検討すべきである．

本研究では Microsoft Windows 向けのポリシー強制エンジンを API フックにて実現したが，もうひとつの実装方法として，秘文 [13] が採用している Windows のフィルタドライバ [*3] として実装する方法がある．制御の網羅性という観点ではフィルタドライバが優れるが，安定性の観点からは OS の外側で制御している API フックのほうが優れており，目的によって使い分けすべきものと考える．ポリシー強制エンジンの OS 依存性を減らすには，例えばコンテンツプロキシとして動作するリファレンスモニタ [14] が提案されている．

[*3] http://msdn.microsoft.com/ja-jp/library/gg155668.aspx

6. おわりに

本研究では，デスクトップクラウドで取り扱われるオフィス文書に対して，その内容に応じたセキュリティラベルを自動的に付与する技術と，オペレーティングシステムに対するアプリケーションの API 呼び出しをフックすることでアプリケーションに許可する動作を制御する技術を組み合わせることで，文書の機密度に応じて制御を動的に変更するデスクトップクラウド向け情報漏洩防止システムを実装した．図 8 は，実際にこの DLP システムを稼働させたデスクトップクラウド環境内で Microsoft Word を起動し，機密文書をポリシーが許していないファイルパスへ別名保存しようと試みて，ポリシー強制エンジンに操作を阻止されている様子を示している．

今後は機密度解析及びポリシー強制の処理オーバーヘッド時間を測定し，今回設計・実装した DLP システムが十分に実用的であることを示す必要がある．また，現実のファイルを収集して，それらが持つ特徴をうまく捉えることができるグラフマイニングのパラメータを特定し，ラベル伝播の精度向上を図る必要がある．

参考文献

[1] The Cost of a Lost Laptop (2009). http://www.intel.com/content/dam/doc/white-paper/enterprise-security-the-cost-of-a-lost-laptop-paper.pdf.

[2] Kashima, H., Tsuda, K. and Inokuchi, A.: Marginalized kernels between labeled graphs, *ICML '03: Proceedings of the Twentieth International Conference on Machine Learning*, AAAI Press, pp.

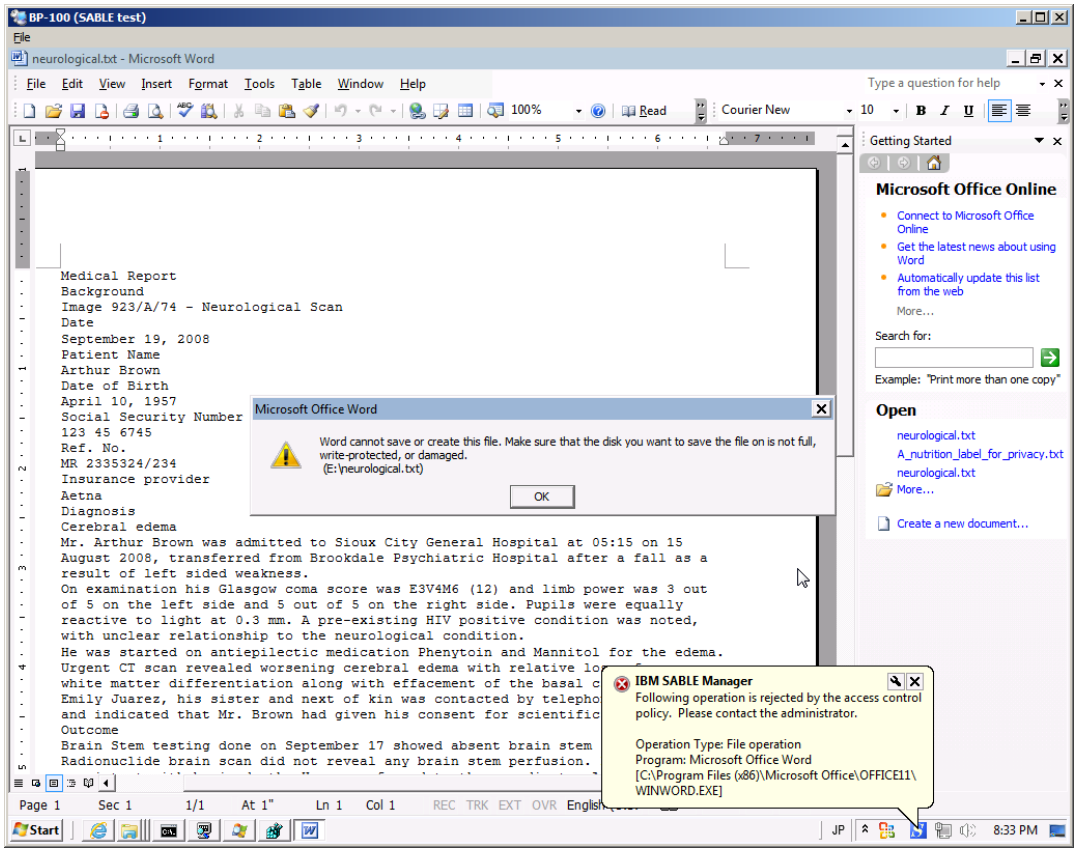


図 8 ファイルの内容に基づいて機密度が判定され，ファイル保存が拒否された例

321–328 (online), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.90.7556 (2003).

[3] Bishop, C. M.: *Pattern Recognition and Machine Learning*, Springer Verlag, New York (2006).

[4] OASIS eXtensible Access Control Markup Language (XACML) TC. http://www.oasis-open.org/committees/xacml/.

[5] 田代 崇，上田高徳，平手勇宇，山名早人：検索エンジンを用いた類似文章検索システム EPCI の評価，*DEWS 2008* (2008).

[6] Metzler, D., Bernstein, Y., Croft, W. B., Moffat, A. and Zobel, J.: Similarity measures for tracking information flow, *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, New York, NY, USA, ACM, pp. 517–524 (online), DOI: http://doi.acm.org/10.1145/1099554.1099695 (2005).

[7] OASIS Open Document Format for Office Applications (OpenDocument) TC. http://www.oasis-open.org/committees/office/.

[8] Office Open XML. http://www.ecma-international.org/publications/standards/Ecma-376.htm.

[9] Wang, Y., DeWitt, D. J. and Cai, J.-Y.: X-Diff: An Effective Change Detection Algorithm for XML documents, *ICDE '03: Proceedings of the 19th International Conference on Data Engineering*, pp. 519–530 (online), http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1260818 (2003).

[10] Viyanon, W. and Madria, S. K.: A system for detecting xml similarity in content and structure using relational database, *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, New York, NY, USA, ACM, pp. 1197–1206 (online), DOI: http://doi.acm.org/10.1145/1645953.1646105 (2009).

[11] Liu, Y., Zhang, D., Lu, G. and Ma, W.-Y.: A survey of content-based image retrieval with high-level semantics, *Pattern Recognition*, Vol. 40, No. 1, pp. 262–282 (online), DOI: http://dx.doi.org/10.1016/j.patcog.2006.04.045 (2007).

[12] Jun Huan, W. W. and Prins, J.: Efficient Mining of Frequent Subgraph in the Presence of Isomorphism, *ICDM '03: Proceedings of the 3rd IEEE International Conference on Data Mining*, pp. 549–552 (2003).

[13] 情報漏洩防止ソリューション 秘文. http://www.hitachi-soft.com/tsg/solutions/hibun/index.html.

[14] Ongtang, M., Butler, K. and McDaniel, P.: Porscha: policy oriented secure content handling in Android, *ACSAC '10: Proceedings of the 26th Annual Computer Security Applications Conference*, New York, NY, USA, ACM, pp. 221–230 (online), DOI: http://doi.acm.org/10.1145/1920261.1920295 (2010).