情報処理学会論文誌　Vol. 50　No. 9　2088–2102 (Sep. 2009)

機密情報の拡散追跡機能による情報漏えいの防止機構

田端利宏[†1]　箱守聰[†2]　大橋慶[†1]
植村晋一郎[†1]　横山和俊[†2]　谷口秀夫[†1]

個人の計算機環境でも，機密情報の管理と漏えいを防止する機構の必要性が高まっている．そこで，本論文では，機密情報が計算機内に拡散する状況を追跡し，機密情報を有する資源を把握する機能，および機密情報が漏えいする可能性を有する処理を制御することにより，未然に情報漏えいを防止する機能を提案する．機密情報拡散の追跡は，ファイル操作，プロセス間通信，およびプロセス生成のシステムコールをフックすることで行う．また，提案方式において，少ない手間で，機密情報の拡散追跡精度を向上させる手法について述べる．事例評価により，機密情報の拡散可能性をすべて追跡でき，利用者の判断を利用することで機密情報の拡散追跡精度を向上させることができることを示す．

Tracing Classified Information Diffusion for Protecting Information Leakage

Toshihiro Tabata,[†1] Satoshi Hakomori,[†2]
Kei Ohashi,[†1] Shinichiro Uemura,[†1]
Kazutoshi Yokoyama[†2] and Hideo Taniguchi[†1]

In personal computer environment, it is important to protect the information leakage. In this paper, a mechanism of protecting information leakage is proposed. This mechanism has two functions. One function is the function of tracing classified information. The other function is the function of controlling the write function. The tracing function is deployed by hook the call of file operations, interprocess communication, and process creation. This paper describes a method that improve the accuracy of tracing classified information and reduce the labor of configuration. This paper shows the proposed mechanism can trace all files and improve the the accuracy of tracing classified information by using the user judgement.

1. はじめに

計算機性能の向上と普及により，様々なサービスにおいて，顧客情報などの機密性の高い情報を計算機で取り扱う機会が増加している．機密性の高い情報は，一部でも外部へ漏えいすると，企業や個人にとって大きな損失になる．また，計算機の世帯普及率の増加により，個人の計算機環境でも，機密情報の管理と漏えいを防止する機構の必要性が高まっている．

個人情報漏えいのインシデントの分析結果[1)] によると，紛失・置き忘れに続いて，管理ミス，誤操作による不注意から起こることが多いことが報告されている．管理ミスと誤操作は，情報漏えい原因全体の約 40%を占めている．近年では，ウィルスやワームが原因で，Winny などの P2P ソフト経由で情報が外部へ漏えいする事例が相次いでいる．特に，組織の情報を自宅に持ち帰って作業をしていた際に，P2P ソフトを通して外部へ組織情報が流出するといった事例が多くみられる．これらの漏えいを防止するには，組織における情報持ち出しの管理を徹底するとともに，個人が利用する計算機環境においても適切なセキュリティ対策を行うことが重要である．

多くのオペレーティングシステム（以降，OS と略す）には，ファイルへのアクセス制御機構が実現されている．しかし，特定ファイルの情報が拡散するのを追跡し，情報漏えいを防ぐ機能はない．また，Security-Enhanced Linux（SELinux）[2)] に代表されるセキュア OS は，計算機上の資源に対するアクセス制御の粒度を細かく設定することにより，必要最小限のアクセス権限をプロセスに与え，不正アクセスにおける被害を抑制できる．しかし，設定工数が大きく，設定したポリシの維持運用にも労力を割く必要があり，導入が難しい．当然のことながら，この対策では，資源に対するアクセス権限を持った者の操作ミスにより起こる漏えいは防止できないという問題がある．

情報漏えいは，プログラムの実行状態であるプロセスが機密情報にアクセスし，計算機外部へ機密情報を伝達することによって起こる．プロセスが情報を伝達する経路には，ファイル操作，プロセス間通信，および子プロセスの生成がある．したがって，情報漏えいの契機を検知し，情報漏えいを防止するには，プロセスの動作により，機密情報が拡散する状況を追跡し，制御することが有効である．ファイル操作，プロセス間通信，および子プロセスの

†1 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University
†2 株式会社 NTT データ
NTT Data Corporation

生成の各処理には，OS が必ず関与している．このため，OS がプロセスの情報や資源へのアクセス状況を監視し，その履歴を取得し，アクセスを制御することで，機密情報を管理できる．

そこで，本論文では，機密情報が計算機内に拡散する状況を追跡し，機密情報を有する資源を把握する機能（以降，機密情報の拡散追跡機能と呼ぶ），および機密情報が漏えいする可能性を有する処理を制御することにより，未然に情報漏えいを防止する機能（以降，書き出し制御機能）を提案する．提案方式では，計算機利用者による管理ミスや計算機の操作ミス，および利用者が意図しない動作をするプログラムが原因となる情報漏えいを防止の対象とする．たとえば，機密情報が格納されたファイルを誤って USB（Universal Serial Bus）メモリにコピーしようとした場合や，電子メールに添付ファイルとして指定して送信しようとした場合がこれにあたる．つまり，計算機利用者が悪意を持つ場合は，対象としない．

提案方式では，計算機の利用者がその計算機の管理者でもある個人が利用する計算機を対象とする．提案方式の利用場面の 1 つとして，個人で管理している計算機において，その利用者が，機密性が高い一部のファイルの漏えいを防止するために，提案方式を利用することを考えている．このため，計算機利用者を信頼できるものと仮定する．今回は，利用者がマルチウインドウで操作できるデスクトップ環境を想定して，書き出し制御機能を設計し，実現した．

提案方式で期待される効果として，設定作業の少なさがある．計算機内には，個人が作成したファイルが存在するものの，その中で外部に漏えいしてはならない重要なファイルは，その一部である．提案方式では，そのファイルリストを作成するだけで，利用を開始できる．

また，提案手法では，機密情報の拡散状況を確実に追跡でき，漏えいの可能性がある場合は，その処理を停止し，警告を表示することで，利用者が許可しない機密情報の漏えいを防止できるという利点がある．機密情報の拡散追跡精度を向上させるには，利用者による判断が必要であるものの，後述するダイアログや機密ファイルリストエディタを用いることで，その作業負荷が少なくなるように工夫している．このため，ポリシを採用したセキュリティシステムのように，事前にすべてのソフトウェアに対するポリシを記述してから，利用を開始するような大きな手間がないことが利点である．

本論文では，2 章で情報漏洩の防止機構について述べる．次に，3 章で機密情報の拡散追跡機能，4 章で書き出し制御機能について述べる．5 章で実現の課題と対処，6 章で実装，7 章で評価結果，8 章で関連研究について述べ，9 章でまとめる．

2. 情報漏えいの防止機構

2.1 基 本 方 式

提案方式の目的は，利用者が意図しない計算機外部への機密情報の漏えいを防ぐことである．この目的を実現するために以下の方式を提案する．

本論文で提案する情報漏えいの防止機構の概要を**図 1** に示す．提案機構は，機密情報の拡散追跡機能と書き出し制御機能を持つ．機密情報の拡散追跡機能とは，機密情報が拡散していく様子を追跡することで，機密情報を持つ可能性のある資源を把握し，機密情報が計算機外へ漏えいする際，それを検知する機能である．書き出し制御機能とは，機密情報の拡散追跡機能が機密情報が漏えいする可能性をともなう処理を検知したとき，利用者へその処理内容を表示し，外部への書き出しを制御する機能である．これらの 2 つの機能により，利用者は，計算機内の機密情報の拡散状態を把握でき，情報漏えいの可能性がある処理の事前確認とその制御を行うことができる．以下に基本的な処理の流れを述べる．

(1) 利用者は，提案機構の起動前，もしくは起動中に機密情報が含まれているファイルを指定できる．

(2) 提案機構は，OS 内部（機密情報の拡散追跡機能）と外部（書き出し制御機能）に実現されており，OS 起動時に各機能は有効になる．

(3) 情報の拡散に関係するシステムコールをフックし，機密情報の拡散追跡機能を呼び出

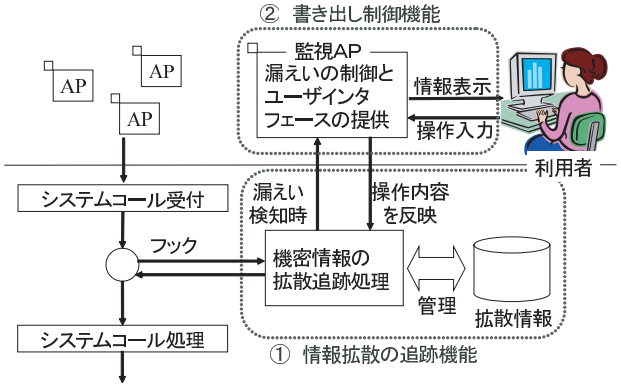

図 1　情報漏えいの防止機構の概要
Fig. 1　Overview of the protection mechanism of information leakage.

す．機密情報の拡散追跡機能では，当該システムコールにおいて，機密情報がどのように拡散するかを把握する．また，必要に応じて，拡散情報の更新を行う．

(4)　上記のシステムコール発行において，計算機外部へ機密情報が漏えいする可能性がある場合には，利用者の判断により，そのシステムコール処理を許可するかどうか決定できる機能を提供する（書き出し制御機能）．また，利用者の判断に基づき，当該システムコール処理を実行するかどうかを制御する．

ファイルに格納されている機密情報は，プロセスを介して，他のファイルや計算機外部へ拡散する．このため，機密情報の拡散追跡機能は，プロセスが機密情報を拡散させる可能性がある操作として，ファイル操作，プロセス間通信，およびプロセス生成において，計算機内における機密情報の拡散の追跡し，その追跡情報を管理する．また，機密情報が計算機外へ漏えいする可能性がある処理を検知した際は，書き出し制御機能に通知する．通知を受けた書き出し制御機能は，利用者に対し，警告とその処理を実行するかどうか，利用者が制御可能なインタフェースを提示し，利用者の判断に基づき制御する．

2.2　機密情報の拡散追跡機能への要求

機密情報の拡散を追跡し，漏えいを検知する機構が満たすべき要件，要望として以下のものがある．

（要件 1）　検知に漏れがないこと
（要件 2）　拡散を即座に追跡可能であること
（要望 1）　検知の結果が的確であること
（要望 2）　処理のオーバヘッドが小さいこと

見逃し（false-negative）をなくし情報漏えいの確実な検知を行うこと（要件 1），および機密情報の漏えい防止のための警告などの制御を即座に行うための追跡法（要件 2）は，機密情報の漏えいを確実に検知するために必要である．

（要望 1）として，的確な検知を行うため，提案機構が機密情報を持つと判断した資源は，実際にも機密情報を所持している必要がある（false-positive の抑制）．また，提案方式を実際に利用するには，提供しているサービスへの影響を抑制し，処理オーバヘッドの抑制（要望 2）が必要である．

2.3　書き出し制御機能への要求

利用者による書き出しの制御を行うことで，漏えいを未然に防止する機構が満たすべき要件，要望として以下のものがある．

（要件 3）　漏えいの可能性を検知した際に，未然に書き出しを確実に止められること
（要件 4）　利用者の判断による書き出しの制御を正確に行えること
（要望 3）　分かりやすい利用インタフェースを提供すること

漏えいを確実に防止するためには，漏えい検知時，その書き出し処理を停止し，（要件 3）利用者による判断を，書き出し処理に正確に反映する必要（要件 4）がある．また，不注意による操作ミスの防止や，GUI 環境での操作しやすさを実現するため，制御機能の操作方法と画面に表示される情報は，分かりやすく，必要最小限であることが求められる（要望 3）．

3.　機密情報の拡散追跡機能

3.1　基本的な情報拡散の経路

以降では，機密情報はファイルの形式で存在し，どれが機密情報に相当するかは利用者が指定していると仮定する．以降では，機密情報を有する可能性があるため，機密情報の拡散追跡機能が拡散情報として管理しているファイルとプロセスを，管理対象ファイルと管理対象プロセスと呼ぶ．

機密情報の拡散は，ファイル形式で存在する機密情報をプロセスが読み込み，さらに他のプロセスやファイルなどへ，その内容を伝えることで行われる．プロセスが情報を拡散する経路を**図 2** に示し，以下に述べる．

(1)　ファイル操作

機密情報が拡散するファイル操作として，ファイル内容の読み込みとファイルへの書

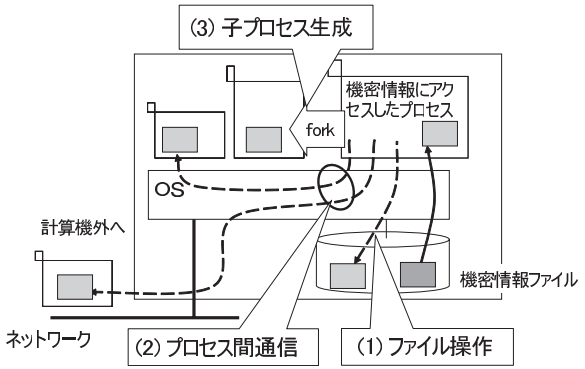

図 2　機密情報の拡散経路
Fig. 2　Diffusion route of classified information.

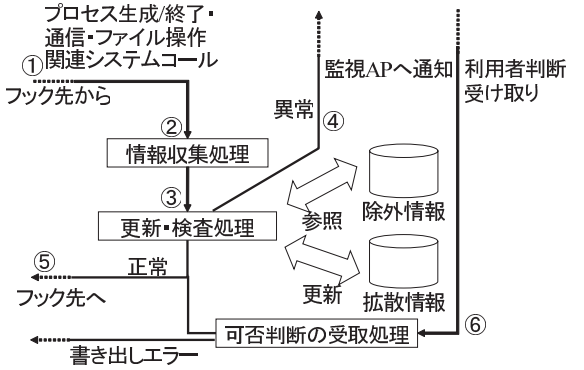

図 3　機密情報の拡散追跡機能の基本機構
Fig. 3　Basic mechanism of the diffusion tracing function of classified information.

き出しがある．読み込み元が機密情報を有するファイル（以降，機密情報ファイルと略す）の場合，プロセスを管理対象とする必要がある．また，管理対象プロセスがファイルに情報を書き出した場合，機密情報が書き出された可能性があるため，書き出し先のファイルを管理対象ファイルとする必要がある．

（2）　プロセス間通信
管理対象プロセスは，プロセス間通信により，他のプロセスに情報を伝達できる．このため，ソケット，共有メモリ，パイプ，およびメッセージキューによるプロセス間通信を監視する．送信元プロセスが管理対象であるとき，送信を仲介する通信用資源と受信プロセスを管理対象とする必要がある．

（3）　子プロセスの生成
子プロセスの生成時に，子プロセスが親プロセスの資源を引き継ぐ機能がある場合（UNIX における fork 処理など）には，この機能を通してプロセス間で情報が拡散する．したがって，親プロセスが管理対象プロセスであるときは，子プロセスも管理対象プロセスとする必要がある．

これらの処理を監視し，機密情報の拡散を追跡する．

3.2　基 本 機 構

機密情報の漏えいを検知する基本機構を**図 3** に示し，以下に述べる．

（1）　機密情報の拡散に関連するシステムコールをフックする．
（2）　情報収集処理では，機密情報の拡散追跡に必要な情報を取得する．
（3）　更新・検査処理では，取得情報をもとに拡散情報を更新するとともに，漏えいの可能性を検査する．
（4）　検査によって機密情報漏えいの可能性（異常）が発見されると，監視応用プログラム（監視 AP）へその旨を通知する．
（5）　検査によって漏えいの可能性がない場合（正常）は，システムコールフック元へ復帰する．
（6）　監視 AP から利用者の判断結果を受け取った後，その判断に従い，システムコール処理を制御する．利用者判断が，実行可能の場合，システムコール処理を継続する．実行不可の場合，システムコール処理をエラーとして終了させる．

上記の処理により，拡散情報を即座に更新でき，漏えいを招く可能性があるシステムコールの実行前に「漏えいの可能性」を検知して監視 AP に通知できる．つまり，（要件 2）を満足している．

4.　機密情報の書き出し制御機能

4.1　設 計 方 針

（要件 3）であげた，計算機利用者の判断による書き出しの制御を正確に行う機能について述べる．1 章で述べたように，提案方式は，計算機利用者による管理ミスや計算機の操作ミス，および利用者が意図しない動作をするプログラムが原因となる情報漏えいを防止の対象とする．このため，書き出し制御機能では，機密情報の拡散追跡機能が機密情報の漏えい可能性を検知したときに，利用者に警告するとともに，漏えいの可能性をともなう処理の実行可否を判断するのに必要な情報を通知する．さらに，利用者がその処理の実行可否を決定できる機能も実現する．

4.2　基 本 機 構

本機能は，OS 外部のプログラムとして実現する．これを監視 AP と呼ぶ．その処理の流れを**図 4** に示し，以下に述べる．

（1）　機密情報の拡散追跡機能からの漏えい可能性検知とその情報の通知を待つ．通知を受け取ると以降の処理を開始する．
（2）　監視 AP は，受け取った情報に基づき，漏えい可能性に関する警告を利用者に表示する．
（3）　利用者は，対象の書き出しの可否を判定し，入力する．

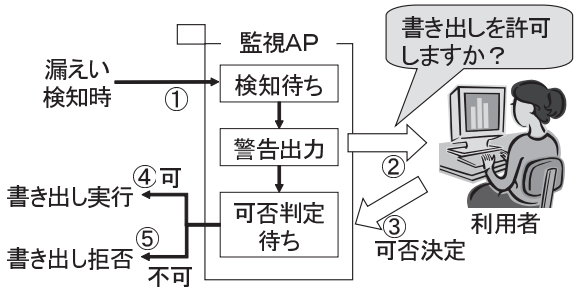

図 4　書き出し制御機能の処理の流れ
Fig. 4　Workflow of the write control function.

図 5　警告メッセージの例
Fig. 5　An example of an alert message.

(4)　書き出しが許可された場合，通常の書き出し処理を行う．
(5)　書き出しが拒否された場合，書き出しエラーを返し，システムコール処理を終了させる．

上記処理により，機密情報の拡散追跡機能による漏えい可能性の検知後，その書き出し処理を中断した状態のまま，利用者の判断に応じた適切な書き出しの制御が可能となる．つまり，(要件 3)，(要件 4) を満たしている．

4.3　書き出し制御インタフェース

利用者への警告の表示内容や，書き出しを制御するための操作が分かりにくいものであった場合，人為的ミスによる漏えいの危険性が高まる．このような漏えいの発生を防止するためには，書き出しを制御するためのユーザインタフェースは分かりやすいことが求められる (要望 3)．このため，監視 AP が利用者に出力する漏えい可能性に関する警告を，GUI によるメッセージで画面上に表示する方式で実現した．利用者に書き出しの可否を尋ねる方式も，警告メッセージウインドウに設置した「はい」，「いいえ」のボタンをクリックするという視覚的に単純な方式とした．

例として，可搬型 USB フラッシュメモリへの書き出し処理において，計算機外部への機密情報漏えいの可能性を検知した際に，監視 AP が表示する警告メッセージを**図 5** に示す．この例は，テキストエディタ emacs が，管理対象ファイルをオープンした後，USB メモリをマウントしたディレクトリ/mnt/usbfm/上の secret.txt というファイルへの書き出しを行おうとした際に表示されたダイアログである．対象ファイルは，機密情報が書き出される可能性があるファイルを示す．プロセス ID とプロセス名は，書き出しを行ったプロセスを識別する情報である．

このダイアログ上で利用者が「はい」，「いいえ」のいずれかのボタンをクリックするとそのボタンに対応する処理が行われる．これにより，利用者は少ない手間で，適切に書き出しを制御できると考える．

5.　実現の課題と対処

5.1　課　　題

提案方式を単純に実現しただけでは，機密情報の拡散を正確に追跡することはできない．これは，機密情報の拡散追跡機能では，検知漏れをなくすために，すべての機密情報の拡散を拡散情報として管理する．これにより，機密情報が実際には拡散していないファイルやプロセスが，管理対象となる．この結果，管理対象のファイルとプロセスのうち，実際に機密情報を保持するファイルとプロセスの割合が低下する．この課題に対する対処を，以降では述べる．

5.2　対　　処

5.2.1　管理対象ファイルリストエディタ

機密情報の拡散追跡機能は，新たに管理対象ファイルとプロセスの情報を拡散情報に追加するのみで，削除する機能がない．そこで，利用者の判断で，拡散情報を確認し，必要に応じてファイルを管理対象から外す機能を実現した．

また，利用者による管理対象ファイルの管理方法として，**図 6** に示す管理対象ファイルリストエディタを提供する．ファイルリストエディタは，管理対象となっているファイルの情報として，フルパス，i ノード番号，ファイルに機密情報を拡散させたプロセス名，およびファイルに機密情報が拡散した日時を表示する．利用者は，ファイルリストエディタを操

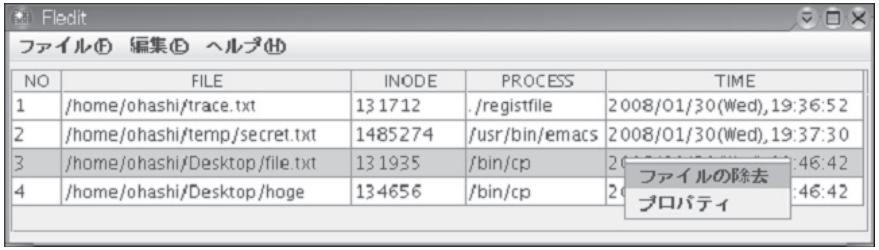

図 6　管理対象ファイルリストエディタ
Fig. 6　A management target files list editor.

作することで，管理対象となっているファイルをリストから除去，もしくは新たにファイルを管理対象として追加できる．

図 6 の例では，ファイルを選択し，右クリックした際に出るメニューを表示している．このメニューで「ファイルの除去」を選択することで，選択したファイルをリストから除去できる．ファイルの編集により機密情報を失ったファイルや，誤って登録された設定ファイルを管理対象から除去することで，管理対象ファイルの増加を抑制できる．

5.2.2　追跡対象からの除外機能

管理対象ファイルリストエディタを用いることで，機密情報を含まないファイルを管理対象外にできる．しかし，機密情報の拡散追跡機能が，機密情報拡散の可能性をすべて拡散情報に追加するため，管理対象ファイルリストを更新する手間がかかる．誤って管理対象ファイルになりやすいファイルの例として，AP の設定ファイルや履歴ファイル，エラー記録用のログファイル（以降，設定ファイルと略す）などがある．また，ファイル操作以外でも，GUI を管理するウィンドウマネージャが誤って管理対象となった際，そのウインドウマネージャと通信を行うすべての GUI プロセスが，管理対象となる．このため，誤った管理対象ファイルの数が急激に増加する．

そこで，AP が起動時や終了時に読み書きする設定ファイルなど，機密情報の拡散が行われないと利用者が判断したファイルとプログラムをあらかじめ監視対象に追加することを除外する機能を追加する．

ファイルについて，以下に述べる．プログラムと，そのプログラムが編集または参照する設定ファイルとの対応を除外情報として指定する．除外情報に指定されたプログラムは，対応付けられた設定ファイルには機密情報を拡散させないものとする．

たとえば，文書編集ソフト emacs と，その設定ファイルである “.emacs-places” を除外情報として指定した際，管理対象プロセス emacs が設定ファイル .emacs-places に書き出しを行っても，.emacs-places は管理対象には追加されない．一方で，emacs 以外のプログラムから .emacs-places を編集した場合，そのプログラムが管理対象であるならば，.emacs-places は管理対象となる．

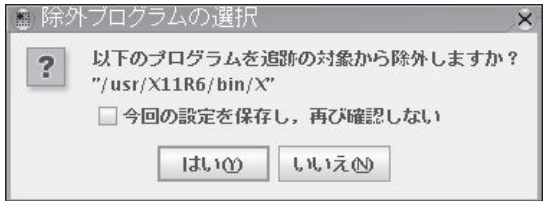

図 7　除外プログラムのメッセージ例
Fig. 7　An example of a notice for wrong management target files.

この手法において，除外情報に指定されたプログラムが対応する設定ファイルに，機密情報を書き込んだ場合，設定ファイルは機密情報を持つにもかかわらず管理対象と見なされない．つまり，false-negative が発生する．これを防止するため，利用者は，プログラムは設定ファイルに機密情報を書き込むことがないと確認したうえで，除外情報の指定を行う必要がある．

次に，プログラムについて述べる．プログラムに対する除外指定は，ウィンドウシステムやウィンドウマネージャが管理対象となることを防止することを想定している．これらのプログラムは，画面表示のために多くのプロセスと通信するため，多くのプロセスが誤って管理対象プロセスとなることが起こる．これを回避するため，利用者があらかじめこれらのプログラムを除外情報として指定する．

除外対象に指定されたプログラム名を持つプロセスは，管理対象として登録されない．具体的な指定方法として，**図 7** に示す選択メッセージを用いる．このメッセージは，機密情報の拡散追跡機能により，プロセスが新たに管理対象となる際に監視 AP によって表示される．利用者は，メッセージに表示されたプログラムを除外対象に指定するか，もしくは指定しないかを選択する．また，チェックボックスにチェックを入れて選択することで，この選択結果が同じプログラム名を持つプロセスに永続的に適用され，以降選択メッセージは表示されない．これにより，利用者がプログラム名を調べて選択する手間と毎回選択する手間を削減し，多数のプロセスと通信プロセスを介して，多くのプロセスが誤って管理対象プロセスとなることを防ぐ．

5.2.3　追跡対象除外ファイルの提示機能

5.2.2 項で述べた追跡対象からの除外機能は，利用者が除外対象ファイルを指定する必要がある．しかし，利用者が，各 AP が読み書きするファイルを特定し，除外情報の指定を行うことは簡単ではない．

そこで，これへの対処として，新たに管理対象となったファイルについて，利用者がそのファイルを管理対象とするか否か判断する手法がある．利用者が，毎回管理対象ファイルとするか判断することで，誤った管理対象ファイルを管理対象から外すことができる．しかし，すべての管理対象ファイル候補に対して，利用者が機密情報の有無を判定するのは，手間がかかり，利便性が低下する．

5.2.2 項で述べたように，誤って管理対象ファイルとなりやすいのは，AP の設定ファイルである．そこで，ファイルが新たに管理対象となった際，そのファイルが設定ファイルの可能性があるかの判定を行う．設定ファイルである可能性が高い場合のみ，そのファイルを除外対象とするかどうかを問い合わせることとする．これにより，除外対象となりやすいファイルを指定する手間を削減することを実現する．

提案方式の実現する Linux では，設定ファイルに傾向がある．

(1)　ファイル名の先頭文字が，‘.’ である．
(2)　AP の実行ファイルがあるディレクトリ，もしくはその下位ディレクトリに保存される．つまり，設定ファイルは，AP の作業ディレクトリとは異なるディレクトリに保存されることが多い．

これらの特徴をふまえ，新たに管理対象の候補となったファイルが，設定ファイルであるかを以下の条件で判定する．

(1)　対象ファイル名の先頭文字が，‘.’ である．
(2)　対象ファイルが属するディレクトリ名と，対象ファイルに書き出しを行ったプロセスの作業ディレクトリ名が異なる．

以上の条件のいずれかにあてはまる場合，対象ファイルは設定ファイルの可能性がある（設定ファイル候補）と見なす．

管理対象に追加されたファイルが条件判定により，設定ファイル候補と見なされた場合，このファイルは誤った管理対象の可能性があるとして，利用者に **図 8** に示すメッセージを表示する．利用者は，対象のファイルに対して，「何もしない（管理対象のまま）」，「管理対象から除去」，「追跡非対象リストに登録（除外情報に指定）」の 3 つの操作を行う．この機能により，利用者は，誤った管理対象ファイルを管理対象から即座に除去を行えるとともに，AP と設定ファイルの対応付けを把握し，除外情報に指定することが可能である．

図 8　設定ファイル候補の通知例
Fig. 8　An example of a notice for configuration files.

5.2.4　対処実現時の利用者の手間についての考察

長期間にわたって提案したシステムを利用した場合，管理対象が少数だったとしても，操作とともに指数的に誤った管理対象ファイルが増加し，無駄な警告メッセージが増えていき，利用者の手間が増大する可能性が考えられる．これに対処するため，提案方式を長期間利用する場合において，その正答率を向上させるための対処を先に述べた．この対処における利用者の手間について述べる．

＜表示内容理解の手間＞

先の対処で述べた各ツールのインタフェースの情報を理解する手間について述べる．

図 5 と図 6 は，ファイル名の指す内容を理解し，プロセス ID，プロセス名，および書き出し時刻から，どの AP（ウインドウ）からの書き出しか把握する手間がかかる．ただし，図 5 の場合，利用者の直前の操作により，メッセージが表示された場合，その特定は比較的容易と考えられる．

図 7 は，プログラムのパス名から，そのプログラムの内容を理解し判断する必要がある．計算機にある程度詳しい人であれば，対象となるプログラムについて判断できる．そうでなければ，正しい判断をするための調査が必要になる．この選択結果は保存できるため，各プログラムに対して 1 度選択するのみである．

図 8 は，各プログラムの設定ファイルか否かを判断する必要がある．これについては，プロセス名とファイル名から，そのファイルについての処理を判断する必要がある．また，設定ファイルの置かれる場所や名前の特徴などを知っていれば，すぐに判断できる．さらに，1 回目の表示で設定ファイルと判断できなくても，AP の処理内容に関係なく繰り返し設定ファイル候補として表示されるファイルであれば，設定ファイルと推察しやすい．

図 9　ポップアップメッセージの例
Fig. 9　Example of pop-up message.

＜長期間利用する際の手間＞

誤った管理対象を除外する作業は，AP の最初の利用開始直後と AP の継続利用時に分けられる．AP の最初の利用開始直後には，5.2.2 項で述べた追跡対象からの除外機能で，機密情報が拡散しないと利用者が判断したファイルとプログラムを監視対象から除外する必要がある．このとき，5.2.3 項で述べた追跡対象除外ファイルの提示機能を利用することで，利用者がすべての除外対象ファイルを調べる手間を削減できる．

次に，継続利用時には，上記の追跡対象から除外できないファイルやプロセスが誤って管理対象となることがある．この対処として，**図 9** に示すポップアップ機能を実現した．ポップアップ機能とは，新たに管理対象が追加されたときに，その情報を随時デスクトップに表示する機能である．管理対象は，利用者の作業とともに追加されることが多い．このため，ポップアップが表示されるときにその内容を確認すれば，利用者が作業した直後であることが多いため，誤った管理対象を削除する際の確認の手間が削減できる．なお，ポップアップ表示時に削除できなかった誤った管理対象は，利用者の判断したタイミングで削除することになる．

提案方式で対象とする機密情報とは，計算機外部に流出してはならない情報である．このため，機密情報の漏洩を防止するためには，ある程度の手間は許容される．また，機密情報が最初から計算機内部に無数に存在することは考えにくい．このため，先に述べた対処を用いて，誤った管理対象の増加を抑制すれば，指数的に利用者の負荷が増加することを防止でき，現実的に利用できる負荷の範囲内で提案システムを利用できると考える．

6. 実　　装

提案方式を Linux 2.6.0 上に実装した．実装においては，ファイル操作と子プロセス生成，およびプロセス間通信により起こる情報の拡散を対象とした．また，情報漏えい検知のタイミングとして，USB で接続された外部デバイス，およびリモートアドレスへの情報の書き出しを対象とした．実装したモジュール構成と管理表を**図 10** に示し，各モジュールの処理概要と管理表について以下に述べる．

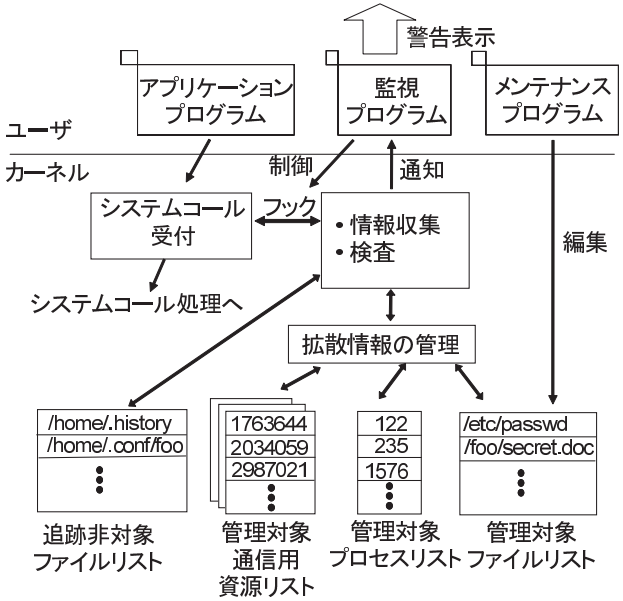

図 10　モジュール構成と管理表
Fig. 10　Modules and management tables.

(1)　システムコール受付
open, read, write, close, fork, exit, msgsnd, msgrcv, send, recv, shmat, shmdt の各システムコールの入り口において，機密情報の拡散追跡機能を呼び出す．

(2)　情報収集と検査処理
3.2 節で述べた追跡手順を実行する．情報拡散に関連するシステムコールが呼ばれると，拡散情報を更新し，経路探索を実行する．ここで，新たにファイルを管理対象ファイルリストに加える際にそのファイルのデバイス番号を確認する．管理対象プロセスがソケット通信を行う際は，通信先のアドレスを確認する．デバイスが外部デバイスであるか，通信先がリモートアドレスである場合，機密情報漏えいの可能性を検知したこととし，監視プログラムに通知する．なお，この処理では，追跡非対象リス

トを参照し，プロセスとアクセス対象ファイルがリストに対応付けられていた際は，拡散情報の更新処理と通知処理は行わない．

(3)　拡散情報の管理
管理対象のファイルとプロセス，および通信用の資源を識別するために，それぞれの情報を格納する管理表を持つ．通信用資源表に関しては，プロセス間通信の種類ごとに用意されている．

(4)　監視プログラム
カーネルから受け取った情報を基に，利用者に漏えいに対する警告や拡散に関しての通知を行う．また，利用者の入力に応じ，漏えいの原因となった書き出しを制御する．

(5)　メンテナンスプログラム
これは，利用者の操作によって，管理対象ファイルリストの内容の編集を行うファイルリストエディタである．

7. 評　　価

7.1 評価目的と項目

提案方式での機密情報漏えい検知時の動作を評価するため，基本評価を行った．次に，機密情報の拡散追跡精度を評価するため，実際の利用事例による機密情報の拡散追跡精度の評価を行った．最後に，処理性能への影響を明らかにするため，提案方式の導入によるオーバヘッドの評価を行った．

基本評価として，管理対象ファイルを読み込み，USB 接続された可搬型ディスクに書き出した場合とネットワークを利用して外部に送信しようとした場合について，評価した．機密情報の拡散追跡精度の評価として，文書作成をともなう 2 つの事例で評価した．また，オーバヘッドの評価として，Linux カーネルのビルド処理時間，および機密情報の拡散追跡機能の影響を受ける各システムコールを評価した．

7.2 基 本 評 価

機密情報漏えいの可能性を検知したときに，正しく外部への書き出しを制御できるか，以下の 2 つの事例で評価した．

(事例 1)　外部デバイス上の書き出しの制御
OpenOffice 3 を用い，管理対象となっているテキストファイルを編集した後，USB 接続された可搬型ディスクへ書き出し処理を行った．このとき，機密情報の拡散追跡機能は，機密情報の漏えいの可能性を検知し，警告メッセージを表示した．警告メッセージの「いいえ」ボタンを押した後，OpenOffice 3 から入出力エラーのメッセージが出力された．これは，検知の原因となった write システムコールの処理がエラーを返したためである．これにより，機密情報の書き出しを未然に防止できた．

(事例 2)　ネットワーク上への書き出しの制御
利用者への通知なしに，管理対象ファイルを読み込み，読み込んだ情報をリモート計算機へと送信するというプログラムを作成し，実行した．このプログラムから生成されたプロセスは，管理対象ファイルを読み込み，管理対象プロセスとなった．このプロセスが，外部へ情報を送信する際，send システムコールをフックした機密情報の拡散追跡機能は，機密情報漏えいの可能性を検知した．警告メッセージに対して，利用者が書き出しを拒否することで，send システムコールの処理がエラーとなり，機密情報の漏えいを未然に防止できた．

7.3 事 例 評 価

7.3.1 評 価 内 容

デスクトップ画面上で実際に作業を行うことを想定して，2 つの文書編集アプリケーションに関して，以下の事例に示す評価を行った．

(事例 1)　外部デバイスへのファイル書き出しの検知
X Window 上で動作するオフィス向け文書アプリケーションである OpenOffice 3 を用い，すでに管理対象となっているテキストファイルを編集した後，内蔵ディスクおよび USB 接続された可搬型ディスクへ書き出したときの検知の結果を比較する．

(事例 2)　複数の文書ファイルに対するアクセス時の検知
KDE が提供するテキストエディタ「KEdit」を利用して，複数のテキストファイルを編集したときの検知結果を比較する．2 つのテキストファイル（ファイル 1，2）を用意し，そのうちファイル 1 をあらかじめ管理対象とする．最初に，ファイル 1 を読み込んで編集し，ファイル 2 として保存する．次に，新たにファイル 3 を読み込み，編集してファイル 4 として保存する．

なお，この評価において，作業はホームディレクトリ下に作成した 1 つのディレクトリ内で行った．また，設定ファイルが管理対象となった際，利用者は 図 8 に示すメッセージ上で，そのファイルを管理対象から除去する操作を行う．

7.3.2 評 価 結 果

評価結果を**表 1** に示す．以下に，表 1 の各項目の意味を述べる．

- 「管理対象ファイル数」とは，利用者が 5.2 節で述べた機能を利用しなかった場合の事

表 1　事例評価の結果
Table 1　Results of case evaluation.

評価項目	事例 1	事例 2
管理対象ファイル数	12	5
設定ファイルと判定された管理対象ファイル数	10	2
利用者の操作を反映した管理対象ファイル数	2	3
機密情報を有するファイル数	2	2
ファイルの網羅率（%）	100	100
利用者の操作を反映した後の正答率（%）	100	66.7
処理中の管理対象となったプロセス数	4	8
処理終了後の管理対象プロセス数	1	0

例評価後の管理対象ファイルの数である．

- 「設定ファイルと判定された管理対象ファイル数」とは，「管理対象ファイル数」のうち，5.2.3 項で述べた機能で設定ファイル候補と判断されたファイルの数である．
- 「利用者の操作を反映した管理対象ファイル数」とは，5.2.3 項で述べた設定ファイルを除外する機能を利用して，利用者が誤った機密情報の拡散を防止した場合の事例評価後の管理対象ファイルの数である．
- 「機密情報を有するファイル数」とは，事例評価後，実際に機密情報の拡散していたファイルの数である．
- 「ファイルの網羅率」とは，機密情報を有するファイルのうち，管理対象とできたファイルの割合である．
- 「利用者の操作を反映した後の正答率」は，利用者の操作を反映した管理対象ファイル数に対するの機密情報を有するファイル数の割合である．
- 「処理中の管理対象となったプロセス数」とは，事例評価中に管理対象となったプロセスの総数である．
- 「処理終了後の管理対象プロセス数」とは，事例評価後に管理対象プロセスとして残ったプロセスの数である．

2 つの事例に対して，機密情報が伝達されたすべてのファイルを管理対象とできた．また，（事例 1）に関して，USB 接続された可搬型ディスクへの書き出しを検知できた．利用者の書き出し制御の操作があるまで，書き出し処理は待ち状態となった．（事例 1）では，利用者は書き出し操作を行っていないため，可搬型ディスクへのファイル書き出し処理は待ち状態のままでである．よって，機密情報を有するファイル数は 2 つしかない．

```
1: Thu Mar 19 13:46:55 2009 253121976 PID: 3065 Process Marked PID:3065 PNM:/usr/bin/emacs FILE:secret.txt
2: Thu Mar 19 13:46:55 2009 1237438015 PID: 3065 SHM Marked PID:3065 PNM:/usr/bin/emacs INODE:753667
3: Thu Mar 19 13:47:39 2009 425406768 PID: 3078 Process Marked PID:3078 PNM:/usr/bin/emacs INODE:753667
   (shared memory)
4: Thu Mar 19 13:48:14 2009 692045424 PID: 3087 Process Marked PID:3087 PNM:/usr/bin/emacs INODE:753667
   (shared memory)
```

図 11　emacs でコピー・アンド・ペーストをともなうファイル操作を行った場合のログの出力例
Fig. 11　An example of log in case of file operations by emacs including copy and paste.

2 つの事例において，設定ファイルと判定された管理対象ファイルは，すべて機密情報を保持していなかった．このファイルを利用者が管理対象から除去，もしくは追跡非対象リストに登録することで，誤った管理対象ファイルの数を削減できる．つまり，機密情報の拡散追跡機能の精度の向上に繋がる．

（事例 2）に関しては，利用者の操作を反映した後も，管理対象ファイル数は 1 つ余分に登録されている．これは，ファイル 2 が管理対象と見なされたためである．このファイルに関しては利用者が，図 6 に示すファイルリストエディタを用いることで管理対象から除去できる．

一方で，（事例 1）は，操作終了後も管理対象プロセスが 1 つだけ残っている．このプロセスは，プリンタサーバの処理を行うデーモン “cupsd” である．このプロセスによる情報の拡散を防止するためには，利用者が手動でプロセスを再起動する必要がある．

7.4　書き出し可否判断に要する手間

情報漏えいの警告メッセージが表示された際に，利用者がその可否判断を行うために要する手間を以下のコピー・アンド・ペーストをともなう処理について述べる．

(1)　emacs で管理対象ファイル A をウィンドウ A で開く．
(2)　emacs で通常ファイル B を管理対象ファイル A とは別のウィンドウ B で開く．
(3)　新しく emacs を別のウィンドウ C で起動する．
(4)　管理対象ファイル A の内容の一部をコピーし，ウィンドウ C の emacs 上にペーストする．
(5)　ウィンドウ C の emacs において，ペーストした内容をファイルとして USB メモリへ書き出す．

これらの手順の操作により，emacs のウィンドウは 3 つ開き，(5) で警告メッセージが出力される．このとき，**図 11** に示す内容がログに出力される．

この例において，利用者にかかる作業の手間を次に示す．

(1)　はじめに，図 5 の形式で表示される警告メッセージから，警告の原因となった AP と

それに対するウインドウを特定する必要がある．このためには，プロセス ID，プロセス名，および書き出し時刻から，ウィンドウを特定する．

(2)　次に，出力されたログから，どのような経路で機密情報が伝達したか調べる必要がある．この例では，図 11 のログの 1〜2，および 4 行目から，管理対象ファイル A の内容が共有メモリを通じて，ウィンドウ A の emacs からウィンドウ C の emacs へ伝達されたことが分かる．

(3)　最後に，書き出すファイルの内容に機密情報が含まれているか否か確認する必要がある．この例では，ウィンドウ C のファイルの内容を確認して，管理対象ファイル A の内容を含んでいるか否かを確認する．

以上のことから，利用者にかかる手間は，漏えいを起こした AP を判断する手間，ログを確認する手間，およびファイル内に機密情報が含まれているか否か確認する手間の 3 つである．

警告メッセージが表示される原因となった AP を判断する手間については，個人のデスクトップ環境では，利用者の操作が契機となり，警告メッセージが表示されることがほとんどであると推察できる．このため，警告メッセージが表示される原因となった AP は，警告メッセージ表示直前に操作していた AP の可能性が高く，この場合の手間は大きくないといえる．一方，不正なプログラムなどが利用者の意志と関係なく書き出す場合には，AP の特定に手間を要すると推察できる．

ログを確認する手間については，機密情報が伝搬した可能性があるログは，プロセス ID またはプロセス名をキーとして，過去にさかのぼって検索する必要がある．このため，ログの量が多い場合は，その手間は増える可能性がある．また，長期的に計算機を利用していると，ログの出力量は累積的に増加するため，これにより，利用者の手間が増える可能性がある．

ファイル内に機密情報が含まれているか否か確認する手間については，目視で書き出し対象ファイルが伝搬した機密情報を含んでいるか否か確認する必要がある．この手間は，書き出し対象ファイルの中の確認すべき情報量に比例する．

7.5　オーバヘッドの評価

7.5.1　システムコール実行時間の評価

提案方式によるシステムコール実行時間のオーバヘッドを評価するため，フックされるシステムコールの処理時間を測定した．測定環境は，CPU が Celeron D 2.8 GHz，メモリが 768 MB の計算機である．

表 2　システムコールのオーバヘッド（μs）
Table 2　The overhead of system-calls.

	機能実装前	機能実装後		オーバヘッド
		非管理対象資源の操作	管理対象資源の操作	
write (file)	21.3	22.4	28.5	7.2
read (file)	6.1	8.0	8.0	1.9
close (file)	3.4	4.5	4.7	1.3
write (pipe)	5.9	6.1	6.9	1.1
read (pipe)	5.2	5.4	5.4	0.2
close (pipe)	5.8	6.5	7.1	1.3
fork	315.4	323.8	337.1	21.7
msgsnd	8.2	8.2	9.2	1.1
msgrcv	6.3	6.6	7.4	1.1
msgctl	5.0	5.2	5.3	0.3
send	2.8	3.6	4.6	1.7
recv	2.6	2.7	2.8	0.2
close (socket)	13.7	15.0	15.6	1.9
shmat	14.1	17.2	16.5	2.3
shmdt	7.0	7.6	7.4	0.4
shmctl	11.0	12.5	12.6	1.5

測定対象のシステムコールは，ファイルの読み書きとファイルディスクリプタの破棄，子プロセスの生成，およびプロセス間通信におけるデータの送受信と通信用資源の破棄を行う各システムコールである．測定は，機能追加前のカーネルと機能追加後のカーネルで行い，処理時間の比較を行った．また，機能追加後のカーネルについては，管理対象と非管理対象のそれぞれの場合におけるファイルの操作やプロセス間通信のオーバヘッドを比較するため，各場合における処理時間の測定を行った．

各カーネルにおける評価結果を**表 2** に示す．表 2 における「管理対象資源の操作」と「非管理対象資源の操作」の項目は，機密情報の拡散追跡機能を実装したカーネルで測定した結果である．また，「管理対象資源の操作」の項目は，管理対象のファイルの操作，もしくは管理対象のプロセスによるプロセス間通信を示す．逆に，「非管理対象資源の操作」は，管理対象でないファイルの操作や管理対象でないプロセスのプロセス間通信を示す．

パイプやファイルの write，fork，send，msgsnd，shmat およびファイルディスクリプタやプロセス間通信の資源を破棄する処理において，1 μs 以上のオーバヘッドがあり，その割合も小さいものではない．その他のシステムコールについては，比較的オーバヘッドが小さ

表 3　bzImage ビルド処理時間のオーバヘッド（ms）
Table 3　The overhead of building Linux kernel.

		実行時間	ユーザ時間	システム時間
機能実装前		348.275	304.367	26.809
機能実装後	管理対象ファイル数 0	349.292	304.797	27.501
	管理対象ファイル数 10	349.878	304.314	27.848
オーバヘッド		1.602	−0.053	1.039

い．これらのオーバヘッドによる AP の性能への影響については，次項で示す．

また，read，msgrcv，recv および共有メモリ操作のシステムコールでは，管理対象資源の操作と非管理対象資源の操作で実行時間に大きな差はない．特に，共有メモリ関連のシステムコールにおいては，プロセスが管理対象であるか否かにかかわらず，同じ内容の処理が行われる．したがって，管理対象プロセスによる共有メモリの操作と非管理対象プロセスによる共有メモリの操作において，大きな差は出なかったものと考えられる．

7.5.2　AP 実行時間の評価

機密情報の拡散追跡機能を導入すると，機密情報を扱っているか否かにかかわらず，すべての read と write システムコールでオーバヘッドが生じる．そこで，提案方式による AP 処理でのオーバヘッドを評価するために，ファイルに対する read と write システムコールを多数発行する Linux カーネルの bzImage のビルドに要する処理時間を測定した．具体的には，機密情報の拡散追跡機能を実装したカーネルにおいて，Linux カーネルの一部のソースコードファイルを管理対象としてあらかじめ登録した状態において bzImage のビルド処理時間を測定する．また，その測定した生成時間を，機密情報の拡散追跡機能を実装していないカーネルでの bzImage ビルド処理時間と比較した．測定環境は，システムコールのオーバヘッドの測定環境と同一である．

評価結果を**表 3** に示す．測定結果は，bzImage のビルド処理時間を 10 回計測した値の平均値である．表 3 における「機能実装前」の項目は，機密情報の拡散追跡機能を実装していないカーネル上での測定値である．また，「管理対象ファイル数 0」と「管理対象ファイル数 10」の各項目は，機密情報の拡散追跡機能実装後のカーネルでの測定値であり，それぞれ，管理対象ファイルをあらかじめ登録しない場合と 10 個のファイルを登録した場合の測定値である．オーバヘッドの項目は，管理対象ファイル数が 10 の場合における機能実装前のカーネルに対するオーバヘッドである．

表 3 から，オーバヘッドの大半は，カーネル内の処理時間に該当するシステム時間であり，ユーザ時間のオーバヘッドはほとんどない．これは，bzImage のビルド中に呼び出されるシステムコールにより，プロセス，パイプ，およびファイルが管理対象になったり，管理対象から外れたりするためと考えられる．実際に，bzImage のビルド処理中における管理対象への登録または管理対象からの削除処理の回数の合計は 147 回であり，短時間で頻繁に登録と削除が行われている．しかし，オーバヘッドは，全体の処理時間に対して 0.5%程度であり，小さい．したがって，本機能の実装による計算機利用者に対する影響は小さいといえる．

8. 関 連 研 究

提案方式のように，情報漏洩の可能性を検知したときに利用者に警告を表示するソフトウェアとして，トレンドマイクロ社のウイルスバスター[3] がある．ウイルスバスターは，クレジットカード番号やパスワードなどの情報を利用者が登録することで，登録した情報が外部の計算機へ送信されるのを検知する機能がある．この機能は，送信データに登録されたフレーズが含まれるかどうかで，検知する．このため，提案手法のように，ファイルそのものを機密情報として登録し，その情報の拡散をとらえて検知することはできない．

情報フローを制御するモデルとして，Bell-LaPadula モデル[4] や，束モデル[5] がある．これらのモデルは，マルチセキュリティレベルにおける情報フローを制御するポリシの提供，定式化を行っている．一方，本研究は，ポリシによる情報フローの制御を目的とするのではなく，情報フローの追跡を行うことで，計算機内における機密情報の拡散可能性を把握することを目的とする．また，利用環境として，上記のモデルが政府システムや軍事システムで利用されることが多いのに対し，本研究は，セキュリティ階層を持たない個人が利用する計算機での使用を想定しており，上記のモデルとは用途が異なる．

情報の漏えいを抑止する方式として，アプリケーションの動作制御や，資源へのアクセス制御を行う方式が研究されている．青柳ら[6] は，機密ファイルを参照した AP の書き出し処理や，プリンタへの出力などの動作を制限することで，情報の持ち出しを防止している．しかし，AP に大きな動作制限が課せられるため，利便性や作業効率が低下してしまう．また，ファイルの持ち出しをすべて禁止してしまう方式は，意図的に機密ファイルを外部へ持ち出す可能性のある個人計算機環境への適用は難しい．喜田ら[7] は，ファイルごとにアクセスを許可するプログラムをホワイトリストに登録し，リスト外のプログラムにアクセス制御を設けることで，ウィルスなどを原因とする漏えいを防ぐ手法を提案している．しかし，この方式はホワイトリストに登録してあるプログラムを利用している際に，利用者の操作ミ

スにより起こる漏えいを防止することができない．鈴木ら[8]は，API フックを用いることで，USB メモリに保存したファイルを USB メモリ外に持ち出すことを禁止する機能を提案している．この研究では，USB メモリに保存したファイルが，USB メモリ外へ漏えいすることを防止する技術を示しており，USB メモリを含む計算機の外部への機密情報の漏えいを防止する本研究とは異なる．

また，カーネル内でシステムコールを補捉し実行の可否を判断する仕組みとして，Krohn ら[9]は，プロセスレベルで情報フロー制御を行う方法を述べている．この研究では，定められたポリシに基づき，プロセス，ファイル，およびプロセス間通信に対してタグを付与し，タグの有無によってファイルの読み書きやプロセス間通信を制御している．

Salvia[10]は，データ保護ポリシとシステムコールの履歴に基づくアクセス制御を OS レベルで実現し，プログラムが動作する環境や状況（コンテクスト）に応じた制御を可能としている．これらの方式はいずれもポリシに従って資源へのアクセスの可否を判断するものであり，情報システムが脅威にさらされる前に対処する点では優れている．しかし，すべての脅威に事前に対応するためのポリシを作成する工数が大きいという問題がある．

古谷ら[11]は，システム上の GUI やプロセスの状態を監視し，それらが変化するたびにアクセス制御ポリシを柔軟に更新することで，画面のスナップショットやコピー&ペーストにより起こる情報フローを制御する手法を提案している．さらに，このポリシを USB メモリ内にデータとともに同梱し，USB メモリ挿入先の計算機にポリシを強制適用することで，USB メモリ内のデータの漏えいを防止することを実現している．この手法は，USB メモリ内のデータの情報フローを制御し，計算機への漏えいを防止するのに対して，本研究では，計算機内のデータの情報フローを追跡し，外部への書き出しを制御する．

Hicks ら[12]は，Jif（Java information flow）言語とセキュア OS の提供する強制アクセス制御を統合することで，OS レベルでのアクセス制御に加えて，AP 内における情報フローの追跡と制御を実現している．しかし，この手法を用いるには，既存 AP をすべて Jif 言語で書き換える必要がある．

利用者の故意や不注意によるデータ漏えいを抑止するには，機密性を要するデータが情報システムから漏えいする時点で検知し対処を行う手法が有効である．これには，データに誰がいつどのようにアクセスしたかを把握することが必要であり，漏えいの可能性が発生した時点で即座に検知し，制御できることが重要である．Goel ら[13]は，ファイルのアクセス処理やプロセス間通信に着目し，計算機内に機密情報が拡散する経路を示す手法が提案している．この手法では，データへのアクセス履歴を詳細に把握することが可能である．しかし，プログラム実行時はシステムコールの発行履歴をデータベースに記録するのみであり，データが漏えいした経路を明らかにするにはアクセス履歴を分析する処理が必要である．このため，データが漏えいする前に検知し抑止することはできない．

鷲尾ら[14]は，ユーザレベルでシステムコールを補足し，ファイルの一部が共有メモリを通して複写された際の機密データの拡散を監視する方式を提案している．ユーザレベルの監視では，カーネル内のリソースの共有や複写によるデータの拡散を検知することができないという問題がある．

情報漏えいを直接抑止するのではなく，計算機上の機密情報の量を減らすことで，情報漏えいの危険性を低下させる研究も行われている．Chow ら[15]は，メモリ上の機密情報の存在期間に着目し，メモリの開放処理時に，開放される領域の内容をクリアすることで，機密情報を取り除く手法を実現している．

9. おわりに

機密情報が計算機外部へ漏えいすることを防止する機構について述べた．提案方式は，機密情報が拡散する経路を追跡するとともに，機密情報漏えいの可能性がある書き出し処理の実行可否を制御することができる．機密情報がファイルで存在することに着目し，機密情報の拡散経路として，ファイル操作，プロセス間通信，および子プロセスの生成をフックし，拡散を追跡する．また，GUI で書き出し制御する機能を実現し，少ない手間で，機密情報の拡散追跡精度を向上させる機能についても述べた．

Linux 上に機密情報の拡散追跡機能と書き出し制御機能を実装して，評価した．利用事例による評価では，提案方式で機密情報が伝達されたファイルをすべてて検知でき，書き出しを利用者の操作により，未然に防止可能であることを示した．さらに，機密情報の拡散した管理対象ファイルリストへの追加と削除処理を，利用者が制御することで，誤って管理対象ファイルとなるファイルやプロセスを大幅に削減できることを示した．また，Linux カーネルの build 処理でのオーバヘッドは，0.5%と少ないことを示した．

残された課題として，機密情報の拡散追跡情報を利用者にとってわかりやすい形式で出力すること，およびより充実した実用的なインタフェースの設計がある．

謝辞　なお，機密情報の拡散追跡機能の検討にご協力いただいた岡山大学大学院自然科学研究科の乃村能成准教授に感謝します．

参 考 文 献

1) 日本ネットワークセキュリティ協会：2007 年度情報セキュリティインシデントに関する調査報告書 ver.1.3 (2008).
2) NSA: Security-Enhanced Linux. http://www.nsa.gov/selinux/ (accessed 2008-11-06).
3) ウイルスバスター．http://jp.trendmicro.com/jp/products/personal/vb2009/index.html（参照 2008-11-30）.
4) Bell, D.E. and LaPadula, L.J.: Secure Computer Systems: Unified Exposition and Multics Interpretation, MTR-2997, MITRE Corporation; ESD-TR-75-306 (1976).
5) Denning, E.D.: A Lattice Model of Secure Information Flow, *Comm. ACM*, Vol.19, No.5, pp.236–243 (1976).
6) 青柳慶光，鮫島吉喜：機密ファイル持出し防止システムの検討，コンピュータセキュリティシンポジウム 2002，Vol.2002, No.16, pp.59–64 (2002).
7) 喜田弘司，坂本　久，島津秀雄，垂水浩幸：ファイルアクセス制御エージェントの提案—P2P 型ファイル共有システムのセキュアな利用を目指して，情報処理学会論文誌，Vol.48, No.1, pp.201–212 (2007).
8) 鈴木大輔，芦野佑樹，佐々木良一：API フックを用いた個人情報漏洩対策システムの提案と多重リスクコミュニケータによる評価，マルチメディア，分散，協調とモバイル（DICOMO2008）シンポジウム論文集，pp.1980–1986 (2008).
9) Krohn, M., Yip, A., Brodsky, M., Cliffer, N., Kaashoek, M.F., Kohler, E. and Morris, R.: Information Flow Control for Standard OS Abstractions, *Proc. 21st ACM SIGOPS Symposium on Operating System Principles* (*SOSP 2007*), pp.321–334 (2007).
10) 鈴来和久，一柳淑美，毛利公一，大久保英嗣：Privacy-Aware OS Salvia におけるデータアクセス時のコンテキストに基づく適応的データ保護方式，情報処理学会論文誌：コンピューティングシステム，Vol.47, No.SIG 3 (ACS13), pp.1–15 (2006).
11) 古市実裕，池部敦巳：セキュリティと利便性を両立するモバイル・オフィス環境の提案，マルチメディア・分散・協調とモバイル（DICOMO2007）シンポジウム論文集，pp.1568–1577 (2007).
12) Hicks, B., Rueda, S., Jaeger, T. and McDaniel, P.: From Trusted to Secure: Building and Executing Applications that Enforce System Security, *Proc. USENIX Annual Technical Conference*, pp.205–218 (2007).
13) Goel, A., Po, K., Farhadi, K., Li, Z. and Lara, D.E.: The Taser Intrusion Recovery System, *Proc. 20th ACM Symposium on Operating Systems Principles* (*SOSP 2005*), pp.163–176 (2005).
14) 鷲尾知暁，除補由紀子，大嶋嘉人，金井　敦：属性の伝播を利用した電子文書の柔軟な利用制御方式の提案，情報処理学会研究報告 2005-CSEC-28，Vol.2004, No.65, pp.375–380 (2005).
15) Chow, J., Pfaff, B., Garfinkel, T. and Rosenblum, M.: Shredding Your Garbage: Reducing Data Lifetime Through Secure Deallocation, *Proc. 14th Conference on USENIX Security Symposium*, Vol.14, pp.331–346 (2005).

(平成 20 年 12 月 1 日受付)
(平成 21 年 6 月 4 日採録)

田端　利宏（正会員）

1998 年九州大学工学部情報工学科卒業．2000 年同大学大学院システム情報科学研究科修士課程修了．2002 年同大学院システム情報科学府博士後期課程修了．2001 年日本学術振興会特別研究員（DC2）．2002 年九州大学大学院システム情報科学研究院助手．2005 年岡山大学大学院自然科学研究科助教授．現在，同准教授．博士（工学）．オペレーティングシステム，コンピュータセキュリティに興味を持つ．電子情報通信学会，ACM 各会員．

箱守　　聰（正会員）

1988 年名古屋大学大学院工学研究科博士前期課程修了．2008 年岡山大学大学院自然科学研究科博士後期課程修了．1988 年 NTT データ通信株式会社（現在株式会社 NTT データ）入社後，オペレーティングシステム，分散処理，情報セキュリティの研究開発に従事．現在，同社ビジネスソリューション事業本部に勤務．博士（工学）．ACM，IEEE-CS 各会員．

大橋　　慶

2007 年岡山大学工学部情報工学科卒業．2009 年同大学大学院自然科学研究科博士前期課程修了．同年シャープ株式会社入社．コンピュータセキュリティに興味を持つ．

植村晋一郎

2008 年岡山大学工学部情報工学科卒業．同年同大学大学院自然科学研究科博士前期課程入学．現在，在学中．コンピュータセキュリティに興味を持つ．

横山　和俊（正会員）

1988 年広島大学工学部第二類（電気系）卒業．1990 年同大学大学院工学研究科博士課程前期修了．2006 年岡山大学大学院自然科学研究科博士後期課程修了．1990 年 NTT データ通信株式会社（現在株式会社 NTT データ）入社後，オペレーティングシステム，分散処理の研究開発に従事．博士（工学）．電子情報通信学会会員．

谷口　秀夫（正会員）

1978 年九州大学工学部電子工学科卒業．1980 年同大学大学院修士課程修了．同年日本電信電話公社電気通信研究所入所．1987 年同所主任研究員．1988 年 NTT データ通信株式会社開発本部移籍．1992 年同本部主幹技師．1993 年九州大学工学部助教授．2003 年岡山大学工学部教授．博士（工学）．オペレーティングシステム，実時間処理，分散処理に興味を持つ．著書『オペレーティングシステム』（昭晃堂）等．電子情報通信学会，ACM 各会員．