社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－OS－103 (11)
2006／7／31

プロセスの通信制限と公開鍵暗号による ファイルの移動制御方式の提案

齋藤 彰一

和歌山大学システム工学部

和歌山市栄谷 930

shoichi@sys.wakayama-u.ac.jp

情報漏洩事件が多く発生している．本提案ではこのような流出を防止する方式について述べる．これらは，ワームや個人の過失によって機密情報を含んだファイルが，第三者が読み出すことができる状態になってしまうことで発生する．提案方式は，1) プロセスが行うファイルアクセスのシステムコールをカーネルによって制御することで移動そのものを制御する．2) ファイルを配布した後に行われる移動 (二次移動) 時に使用する公開鍵をファイル作成者があらかじめ指定する．これら 2 つの方式を併用する．これらによって，ファイルの一次移動と二次移動の対象を制御する．本論文では，提案方式の詳細と，Linux カーネルに構築しているプロトタイプシステムの概要について述べる．

Proposal for File Transfer Control System with Limited Commnunications and Encrypting by Public Key Cryptography

Shoichi SAITO

Faculty of Systems Engineering, Wakayama University

930 Sakaedani, Wakayama JAPAN

A lot of information leak accident has been reported. In this proposal, a new method to prevent an information leak accident is described. The accident is caused by that third parties come to be able to read a file including sensitive information by worms and human errors. This proposal method is composed of the following two steps. As the first step, the kernel limits execution of system calls related to a file access. As a result, the file transfer is controlled by the proposal system. As the second step, the file creator specifies the public key for encrypting the file. If a file receiver transfers the received file to third parties, the proposal system encrypts it using the public key. The third person can't come to read it. In this paper, details of the proposal method and the outline of the prototype system which is constructed on the Linux kernel are described.

1 はじめに

情報流出防止のために，個人情報などの機密情報を含んだファイルを外部へ持ち出すことを禁止する企業が一般化している．しかし，顧客回りなどの営業や出張のためにファイルを外部持ち出す必要性がなくなったわけではない．そこで，本論文では，安全にファイルの配布と移動を行う方式についての提案を行う．

ファイルの持ち出しが困難な状況は，企業活動にさまざまな影響を及ぼしている．情報流出を防ぐためにファイルの持ち出しを禁止する厳しい社内ルールがかえって情報流出を招いているという分析 [1] もある．このため，ファイルの持ち出しや持ち出し後のファイルをネットワークで追跡する監視システムが発売されている [2]．このような製品では，持ち出し制限やアクセス制限，ログによる編集や移動の監視を行うことができる．しかし，システムには監視用のサーバの設置が必要となるなど運用面での負荷がある．特に，中小企業や個人での利用は難しいと思われる．

ファイルを安全に持ち出すための条件を考える．1)

正当な権利を持たない者によってファイルを移動させない．2) 配布されたファイルは指定された個人 (組織) 以外は読めない．3) 正当に配布されたファイルが，第三者に勝手に再配布されない．これら条件の内，1) は自計算機による管理によって制御できる．2) はファイルの暗号化によって実現できる．3) は上記製品などを用いれば可能である．しかし，より簡易な，例えば監視サーバを用いない個人向けの実現方法はない．

本論文では，コンパイラによるプログラムの解析とカーネルのプロセス監視によるファイル単位の移動制御機能と，カーネルによる公開鍵暗号を用いたファイルの暗号化を組み合わせた方式を提案する．本提案システムの特徴の一つ目は，公開鍵暗号の鍵配送に PKI を用いることである．これにより，監視・管理サーバを設置する必要がなくなり，低コストでの運用ができる．PKI に住民基本台帳カードを活用できれば，PKI の運用コストもなくすことができる．特徴の二つ目は，配布した後の移動制御を可能とし，さらに，ファイル作成者が指定した先には二次移動を可能とする点である．これにより，ファイル作成者との間のファイル移動や，ファイル作成者がファイルをノート PC 等の携帯端末に保存して外部に持ち出すことを，流出の危険性なしに可能にする．

本提案方式は，条件 1) から 3) のすべてをクリアする．1) については，プロセスが行うシステムコールを監視し制御することによって実現する．2) はファイル単位の暗号化を PKI による公開鍵暗号方式を用いて実現する．3) は，ファイル単位で二次移動可能対象の指定を行い，さらに二次移動時に公開鍵暗号方式を用いて強制的な暗号化を行うことで実現する．本論文では，提案の詳細と，Linux カーネル上に実現したプロトタイプについて述べる．

2 ファイル単位での移動制御

本システムが提案するファイル単位で情報を保護することの必要性について述べる．さらに，ファイル単位での保護をカーネルによって実現するために必要となるコンパイラとの連携について述べる．

2.1 ファイル単位での保護の必要性

本論文では「計算機からの情報漏洩」を，『機密情報を含むファイルを，本来は閲覧を許可されていない第三者が閲覧可能となる状態』と定義する．このような情報漏洩を未然に防止するには，計算機の認証やファイルの暗号化，移動時の暗号化通信は当然のことである．しかし，これだけでは完全ではなく，さらにワームやセキュリティホールを介した攻撃によるファイルの無断送信，利用者の過失による送信も考慮しなければならない．最近では，このような保護のために，SELinux[3] を用いた運用が行われるようになった．しかし，SELinux において適切な運用が行われた場合においても，利用者の過失のような，正当なアクセス権を有するプロセスからファイルを保護することは難しい．この場合においてファイルを保護するためには，プロセス単位の制御に加えて，個々のファイル単位の制御を行う必要がある．

また，ファイル単位での保護には情報の粒度をファイル単位まで細かくすることができるという利点がある．職場などで計算機やネットワークの利用目的を限定し，FireWall などを用いて情報の移動を管理することがある．この場合，ネットワーク内がすべて同一のセキュリティポリシーである必要がある．もし，異なるセキュリティポリシーに属する情報が含まれる場合，その情報に対しては特別な制御機構が必要となる．しかし，ネットワークによる制御では，情報の移動制御の最小粒度がネットワーク単位となる．つまり，ネットワーク単位での管理においては，個々の計算機内部にセキュリティポリシーが異なる情報が含まれていた場合，それぞれの情報に対して独立した保護を行うことが難しい．一方，ファイル単位による制御であれば，その制御の粒度をファイルまで細かくすることができる．人による情報交換はファイル単位によって行われることが多い．よって，ファイル単位での保護機能が有用である．

2.2 カーネルによる移動制御

計算機内外でプロセスによるファイルの移動 (拡散) がどのように行われるかについては文献 [4] に詳しい．簡単に述べれば，機密情報を含むファイルを読み出したプロセスが行うファイル操作による書き込み，プロセス間通信によるパケット送出，子プロセスの生成がファイルの移動を引き起こす．

上で述べたファイル移動は，OS にとってはシステムコール単位でしか把握することができない．つまり，機密情報を読み出したプロセスが別ファイルに書き込んだ内容を，機密情報か否かを判断することは OS にとって困難である．これは，一旦プロセス内部に読み出されたデータが，プロセス内部でどの

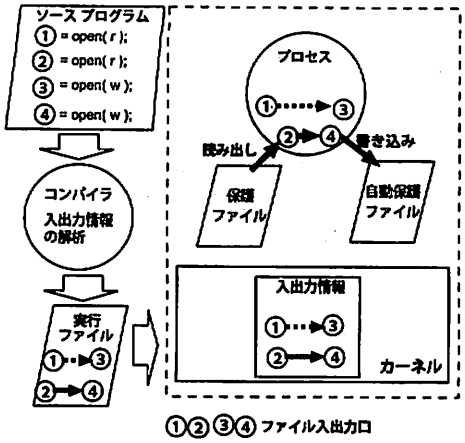

図 1: コンパイラによる入出力解析とカーネルによる移動制御

ように参照，書き換え，加工されているかについて，プロセス外部にあるカーネルが確実に検出することは困難なためである．よって，ファイルから読み出された機密情報が，どこに出力されるかについて確実な情報をカーネルは知ることができない．このため，機密情報を含んだファイルを一度でも読み出したプロセスが生成するファイルは，機密情報が書き込まれる可能性があるためにすべて保護する必要があることになる．これは現実的ではない．例えば，エディタプログラムの設定ファイルに機密情報が含まれると間違って設定された場合，その後に編集するすべてのファイルに機密情報が含まれる可能性があると判断される．このため，編集するすべてのファイルが保護対象となってしまう．よって，本提案システムでは，コンパイラによってプロセス内のデータの流れを把握し，その情報を基にしてカーネルによるファイルの入出力制御を行う．このような機能を持ったコンパイラの研究として，言語レベルでの情報流の把握を目的とした VITC[5] などがある．

コンパイラを用いた情報流解析とカーネルによる移動制御の関係を図 1 に示す．図 1 では，入出力口 (ディスクプリタ) の 1 番と 3 番，2 番と 4 番がそれぞれ関連付けられているとコンパイラが解析できた例を示している．また，コンパイラの解析結果は目的コードに埋め込まれる．そして，プロセスが入出力口の 2 番から保護されたファイル (保護ファイル) を読み出した結果，対応する 4 番から出力したファイルが自動的に保護されたファイル (図 1 の自動保護ファイル) になっている例を示している．

3 移動制御と保護の概要

本提案システムの前提条件と，ファイル移動制御の概要およびその手順について述べる．

3.1 前提条件

本提案システムの安全を確保するための前提条件を以下に示す．

- 暗号は安全とする．
- 公開鍵方式と PKI などによる鍵配送システムは安全とする．
- Linux カーネルと SELinux は安全とし，設定も適切とする．暗号解読のためのカーネルの改変などは考慮しない．
- ハードウェアは安全とする．ハードウェアキーロガー，ディスプレイからの電磁波などハードウェアに依存するセキュリティ保護は対象としない．
- 印刷物や画面への出力等の出力結果は保護の対象としない．

3.2 提案方式の概要

本提案方式は，管理サーバを使用せずに，配布を受けた者からの移動 (以下，**二次移動**と言う) を制御する．このために，保護すべきファイルにラベルを設定し，ファイル単位での保護を行う．以下，ラベルを設定されたファイルを**保護ファイル**と言う．保護ファイルは，ラベル設定と同時に暗号化[1] して二次記憶に保存する．ラベルには移動対象や方法等を記す．以下，システムの全体構成と，アクセスの種類によるアクセス処理方式について述べる．

3.2.1 全体構成

提案システムの全体構成を図 2 に示す．本システムは，ファイル作成者が，保護ファイルに 1) ラベルを設定することから始まる．ラベルには，移動方法や二次移動先公開鍵等を指定する (詳細は 4 章で述べる)．保護ファイルが移動する場合には，2) 移動先の公開鍵によって暗号化される[2] (詳細は次項以降で述べる)．また，3) ラベルの内容と保護ファイルへの

[1] 利用者別の共有鍵を用いる．
[2] 移動先が複数ある場合は，個々の公開鍵によって暗号化する．

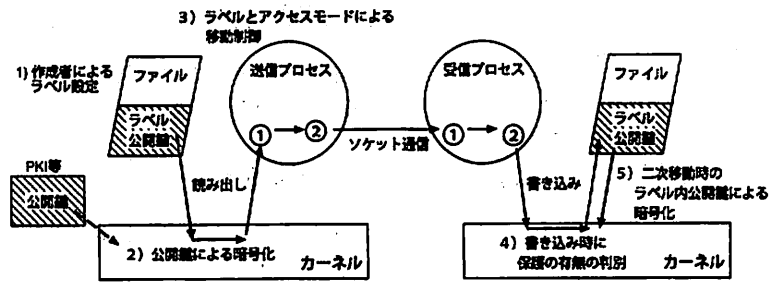

図 2: 提案システムの全体構成

アクセスモードによってプロセス内での移動制御を行う．2) と 3) により，他のメディアへの保護ファイルを書き込む場合に，暗号化と移動先の制限を行う．つまり，保護ファイルは移動時には必ず暗号化されていることになる．

移動先プロセスがファイルを受信した場合，当該ファイルは公開鍵で暗号化された状態である．これを HDD などのメディアに書き込む際に，4) カーネルは埋め込まれたラベルを自動検出し，以降，保護ファイルとして扱う．ラベルは利用者の鍵を用いて復号され，プロセスの移動制御に用いる．最後に，二次移動を行う場合には，5) ファイルの暗号化を，ラベル内に埋め込まれた二次移動対象公開鍵を用いてカーネルが行う．これにより，二次移動後のファイルは，ファイル作成者が指定した者 (二次移動対象公開鍵に対応した秘密鍵を有する者) のみが読み出すことができる．以上により，移動先におけるファイルの保護と，二次移動の制御を行う．

3.2.2 移動の有無によるアクセスの種類

本方式は，保護ファイルを読み出すプロセスが，ファイルを移動させる (他のメディアに書き込む) 場合と，移動させない場合において，アクセスモードを区別する．移動を伴うアクセスでは，移動先に対応した保護ファイルの暗号化をカーネルが行い，プロセスに暗号化した状態でファイルを読み出させる．一方，移動を伴わないアクセスの場合には，流出の危険性がないことから，カーネルが復号し，平文による読み出しを行う．これら 2 種類の区別は，open システムコール時に判断する．移動を伴うアクセス時には暗号化に対応するために copen (crypted open) システムコールを新設する．また，移動を伴わないアクセス時には通常の open システムコールを用いる．これらの区別の判断は，2.2 節で述べたようにコンパイラによる支援を得て判断を行う．しかし，現状では区別することができないため，アプリケーションプログラマの判断によって行う．

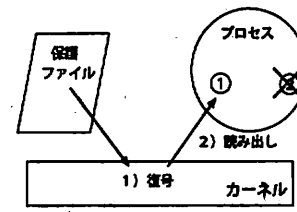

図 3: 移動を伴わないアクセス処理

3.2.3 移動を伴わないアクセス

移動を伴わないアクセスの場合，open システムコールによってファイルをオープンする．処理の概要を図 3 に示す．平文による読み出しを行うため，カーネルは，ファイル所有者の鍵を用いてファイルを復号しプロセスにデータを渡す (図 3 の 1)．また，他のメディアに対する書き込みは許可されない (図 3 の入出力口の 2 番)．なお，この処理は，ファイル作成者においてもファイルを受信した者においても同じである．

3.2.4 移動を伴うアクセス

移動を伴うアクセスの場合，新設する copen システムコールを用いてファイルをオープンする．この場合，移動のためにファイルを暗号化するが，ファイル作成者による移動 (一次移動) と二次移動によって暗号化に用いる鍵が異なる．一次移動の処理の概要を図 4 に，二次移動の処理の概要を図 5 にそれぞれ示す．まず，一次移動の場合でも二次移動の場合でも，カーネルがファイル所有者の鍵によって暗号化された保護ファイルを復号する (図 4,5 の 1)．次に，

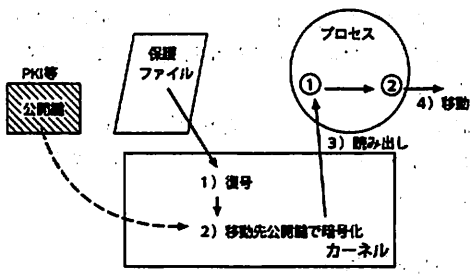

図 4: 一次移動時のアクセス処理

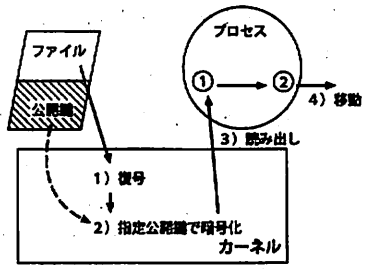

図 5: 二次移動時のアクセス処理

一次移動の場合は，カーネルは，ファイル移動先の公開鍵を PKI 等の鍵配送基盤を用いて入手し (図 4 の点線)，その公開鍵でファイルを暗号化する (図 4 の 2)．プロセスの読み出しは，移動先の公開鍵で暗号化された状態で読み出される (図 4 の 3)．これによって，プロセス内部を含めて移動先以外での復号ができないように保証する．

二次移動の場合も公開鍵を用いて暗号化を行うが，公開鍵の入手方法が異なる．先に述べたように一次移動では，PKI 等の鍵配送基盤を用いるが，二次移動では作成者が一次移動時にラベル内に埋め込んだ公開鍵を用いる (図 5 の点線)．つまり，二次移動では，移動先をファイル作成者が指定し，移動を行う者 (ファイル作成者より保護ファイルを受け取った者) は移動先を選ぶことはできない．これにより，二次移動先をファイル作成者が制御することを可能にする．結果，ファイルを受け取った者がワームなどに感染しても，第三者にファイルを送出されることがなくなる．

3.3 システム構成

前節までに述べたシステムを実現するために必要となるシステム構成について述べる．まず，コンパイラによる情報流の解析が必要である．次に，解析結果を安全にカーネルに伝える仕組みが必要である．これは，実行ファイルの形式を拡張し，解析結果の埋め込みと作成者による署名を行うことで実現する．最後に，プロセスを実際に制御するカーネルが必要である．また，既存技術としては，PKI や IC カードなどの鍵管理基盤が不可欠である．以下，開発対象となるコンパイラとカーネルの概要について述べる．

3.3.1 コンパイラ

コンパイラは，アプリケーションプログラム全体を解析し，入出力に関係するシステムコールを抽出する．次に，各入力口から読み出したデータが，プログラム内の各変数 (メモリ空間) を経由して送られる出力口を特定する．実行ファイル生成時に，この入力口と出力口の組を実行ファイル内部 (実行ファイルの形式の拡張部) に格納する．入出力にはファイルとソケット以外にも mmap などの方法もあるが，現在，コンパイラおよび実行ファイルの拡張の仕様については検討中である．

3.3.2 カーネル

カーネルは，プロセス生成時にコンパイラが解析した入出力口の組を受け取り，プロセスの移動制御情報に加える．これを用いて，ある入力口から保護ファイルが読み出された場合，コンパイラによって関連が検出された出力口からの出力ファイルには，入力ファイルと同様のラベルをカーネルが自動的に設定する．これによって，プロセス内部のデータ移動に関して，従来のブラックボックス的な扱いよりも，高精度な保護が実現する．

4 移動制御ラベル

個々のファイル単位で移動制御を行うためには，ファイル単位に移動制御の判定の基となる情報を設定する必要がある．本提案ではこの情報を移動制御ラベル (Transfer Control Label. 以下，TCL) という．

TCL は，ファイル作成者が個々のファイルに設定する．また，open システムコールを通じて，ファイルにアクセスしたプロセスにも設定されてファイル移動制御の基となる．TCL の構成を以下に示す．

- ファイル作成者情報
- 二次移動対象 (組織，個人) の公開鍵
  二次移動の制御に用いる．

- 移動方法・範囲の指定
  ファイル作成者に対する一次移動の制御に用いる．
- 作成者による署名

ファイルの作成者は，そのファイルに対する移動制御を行う権利を有する者であり，移動対象者の設定や変更をすることができる者である．ファイル作成者情報とは，ファイル作成者を特定するための情報であり，作成者指定の情報を暗号化したデータである．作成者情報は，作成者のみが知りうる情報 (パスワードや PIN 等) を用いてのみ復号可能である．TCL の初期化時に TCL に埋め込む．TCL 変更時には，この作成者情報が正しく識別できることで作成者を認証する．

二次移動対象の公開鍵は，ファイル配布後の二次移動を制御するために用いる．一般に，ファイルを一旦配布した後のアクセス制御は困難である．市販システムでは，専用アプリケーションを利用して，アクセスのための鍵を管理サーバから入手することで配布後ファイルへのアクセス制御を行う．本システムでは，ネットワークを用いずにカーネルのみで制限を行う．本システムのカーネルは，他者から TCL 設定で送られてきた保護ファイルをさらに移動させる場合，TCL に指定された二次移動対象公開鍵によってファイルを暗号化する．これにより，二次移動後のファイルは，二次移動対象公開鍵に対応した秘密鍵でのみ復号可能となる．つまり，ファイル作成者は，一次移動後の二次移動先の公開鍵を TCL に埋め込むことにより，二次移動先を制御，限定することができる．例えば，移動許可先にファイル作成者自身の公開鍵を埋め込むことで，二次移動先を作成者自身に限定できる (作成者と配布先との単純移動のみ可能となる)．このようにして，ファイルの移動を制御する．

三点目の，移動方法と範囲の指定は，ファイル作成者がファイルを移動させることができる範囲とその方法を制御するために用いる．一般の組織では，ファイルの管理権限はファイル作成者よりも所属する組織にある．よって，ファイル作成者がファイルを移動させる範囲も組織によって管理される場合が多い．その管理に本指定を利用する．本指定では，ファイルを書き込むことができるデバイスや，ネットワークを介した通信先を指定する．本指定の設定は，組織のセキュリティポリシーに依存するため，ファイル作成者や端末利用者ではなく，組織のシステム管理者が各端末に設定する．なお，この範囲や方法以外での移動が必要となった場合には，権限がある者によって特別な許可をあたえることによって移動を可能とする．

最後に，TCL 全体を作成者の秘密鍵によって電子署名を行う．これによって，TCL の内容が正しいことを確認する．

5 保護の例

本システムによるファイル保護について，ワームによるサーバプロセスの乗っ取りと，利用者の間違いにより保護ファイルをメールに添付しようとした場合を例に述べる．

5.1 ワームによるファイル移動からの保護

WWW サーバなどインターネットからの接続を受けるサーバプロセスにおいて，そのプロセスのセキュリティホールなどを介してワームが感染した場合，ファイルなどを無作為にアクセス，ネットワークに送出することがある．本システムを利用した場合には，ワームの感染そのものを防ぐことはできないが，ワームに感染したプロセス (以下，ワームプロセスという) が保護ファイルを移動させることを防止ことができる．つまり，ワームプロセスが open システムコールを使って保護ファイルを読み出すと，カーネルによって，ワームプロセスは通信関係のシステムコール (socket 等) を実行できなくなる (3.2.3 項参照)．また，すでに socket を開いていた場合には，保護ファイルに対する open システムコールが失敗する．これらは，カーネルによって強制的に行われるためにワームプロセスと言えども逃れることはできない．

また，ワームが保護ファイルを copen システムコールを使って暗号化のままの読み出しを試みた場合には，ファイル作成者のみが知り得る情報 (パスワード等) が必要となり，通常読み出しは失敗する．しかし，利用者がパスワードをメモリやファイルに記憶しており，ワームがそれを利用した場合には，読み出しが成功して通信可能となる．しかし，この場合においても，カーネルによって強制的に TCL 内部の公開鍵による暗号化が行われ，第三者が復号可能な状態にはならない．以上から，ワームによるファイル移動からは安全に保護される．

5.2 利用者の過失によるファイル移動からの保護

利用者がメールにファイルを添付する際に，本来の添付すべきファイルと間違って保護ファイルを添付する場合を考える．この場合，保護ファイルが添付されることを防ぐことはできない．メーラが添付されるファイルが正しいか否かを判断することはできないからである．ここでメーラが本システムに対応したコンパイラでコンパイルされていない場合は，通常の open システムコールによる保護ファイルの読み出しとなり，メーラには TCL が適用される．これために，通信関係のシステムコールを実行できない．よって，保護ファイルがメールに添付されて送られることはない．また，対応コンパイラによるコンパイルが行われていた場合や copen システムコールを用いた暗号化による添付ファイルの読み出しに対しても，ワームの場合同様に，カーネルによる強制的な暗号化により第三者が復号できない状態となる．

6 プロトタイプの実装

本提案を実現するために，カーネル部分のプロトタイプを実装した．現時点では，ファイル単位での保護の有無のみを指定する簡易 TCL の設定，保護ファイルのアクセス時に他メディアへの書き込みの禁止，ネットワーク送信の禁止が可能である．なお，ファイルの暗号化は実装されていない．また，保護ファイル書き込み時の自動的 TCL 継承機能は限定された機能のみが実装されている．

6.1 ファイル保護の継承

現時点では，ファイル入出力の解析に対応したコンパイラは存在しないため，保護ファイルを読み出したプロセスが行うすべての書き込みに機密情報が含まれると仮定しなければならい．しかし，2.2 節で述べたように，すべての書き込みに簡易 TCL を適用した場合には，無用な保護の伝播が発生する．このため，保護ファイルを読み出したプロセスのファイル書き込みによる簡易 TCL の伝播には，以下のポリシを用いて簡易 TCL の設定の有無を判断している．

- 新規作成ファイルには簡易 TCL を適用する．
- 当該プロセスが unlink もしくは rename したファイルと同じ名前のファイルを新規作成した場合には，元のファイルと同じ簡易 TCL を適用する (ファイル名による簡易 TCL 有無の継承)．
- ファイル内容の書き換えには簡易 TCL を変更 (追加・削除) しない．

このポリシでは，保護ファイルを更新した場合は，簡易 TCL はそのまま継承される．また，保護ファイルを読み出し後に作成されたファイルにも簡易 TCL が設定される．しかし，保護ファイル読み出し以前より存在した簡易 TCL のないファイルへ書き込みを行った場合には，簡易 TCL は設定されない．現時点ではここから機密情報の流出可能性がある．

6.2 実装方法

プロトタイプシステムにおいて実装した，提案システムの機能と実現方式について述べる．

6.2.1 簡易 TCL の保存

プロトタイプでは，簡易 TCL を inode 内の拡張属性 [6] に記録している．簡易 TCL の内容は，保護の有無のみである．記録用に settranscntl システムコールと gettranscntl システムコールを追加した．それぞれ，簡易 TCL の設定と状態の取得を行うシステムコールである．

6.2.2 ファイル open 時の処理

open システムコールと socket システムコールにおいて，当該ファイルおよびそれまでに open したファイル (当該プロセスのディスクプリタ 3 番以上) について簡易 TCL の有無を確認し，open および socket システムコールの判定を行っている．もし，保護ファイルであった場合には，プロセスに簡易 TCL を付加する (プロセス構造体に記録する)．これは，保護ファイルの close 後も保護状態を保持するためである．

6.2.3 ファイル置き換え時の処理

ファイル名と inode は独立した存在であるため，プロセスがファイルを置き換えると，同一ファイル名ではあるが inode は異なる状態になる．このために，元のファイルに設定されていた TCL が無くなる．しかし，ユーザにとっては同一のファイルであるために，inode に基づくだけではなく，ファイル名による TCL 有無の継承が必要である．これを行うために，unlink システムコールと rename システムコールに

おいて，削除したファイル名のハッシュ値などをプロセス構造体に記録している．逆に，ファイルを作成するrenameシステムコールとopenシステムコールでは，当該プロセスが削除したファイルと同名のファイルを作成した場合には，自動的にTCLを付加する．

7 TCLの保護

本システムにおいてファイル保護に重要な役割を果たす物がTCLである．TCLの有無と内容によって保護が実施される．よって，ファイル作成者の意図しない削除や改変が行われないように，TCLを保護しなければならない．TCLは，二次記憶等に保存されている状態においてもネットワークで移動中においても，ファイルの一部に埋め込まれて暗号化される．このため，暗号が充分な強度を持っていれば基本的な保護は可能である．ただし，TCLを埋め込む形式を解読しにくいように工夫する必要などがある．また，プロトタイプ実装では，inodeの拡張属性領域に保存されている簡易TCLを，他のプロセスより保護する必要がある．OS標準機能を使用しているため，OS付属のgetfattrコマンドなどによって容易に改変が可能である．よって，TCL利用時には署名の確認をそのつど行うなどの対策が必要である．

なお，TCLの変更は，ファイル作成者以外はできない．これは，1)TCLの変更にはファイル生成者情報(4章参照)によって確認が行われる．2)TCLにはファイル生成者の署名が必要である．変更時には再署名が必要であるが，一般に作成者の秘密鍵は入手できないため署名を偽造することは困難である．以上によってTCLの保護を実現する．

8 関連システム

本システムは，他のセキュアOS，特にSELinuxとの共存によりその安全性を保証する．本システムは，ファイルの移動を制御するのみで，プロセスに必要最低限の権利を与える最小特権機能などを実現するものではない．例えば，ワームプロセスがなんらかの手段でファイル作成者の情報を入手して，TCLを削除するシステムコールを発行した場合には，それを拒否することはできない．よって，SELinuxなどを用いて，事前にサーバプロセスなどへの最小特権(TCLに対するシステムコール発行の禁止)を設定する必要がある．この上で，本システムは，個々のファイル単位の移動を制御するシステムである．よって，本システムは，SELinuxなどと相対するものではなく，補完しあう関係にあるといえる．

9 おわりに

機密情報を含んだファイルを意図しない移動から保護するためのシステムについて，公開鍵暗号とプロセスの制御による方式について述べた．さらに，Linuxに構築しているプロトタイプシステムについて述べた．本システムは，ファイル移動のための監視サーバを使用せずに，PKIによる暗号化によって移動の制御を行っている．これにより，監視システムを導入できない中小企業や個人でも容易にファイルの移動制限を行うことができる．

本システムの安全性は，コンパイラによるプロセスの入出力情報が不可欠である．よって，今後，コンパイラの仕様の検討を行う計画である．また，プロトタイプシステムを拡張し，ファイルの暗号化と公開鍵方式への対応を行う計画である．

謝辞 本システムを設計するにあたり，貴重な御意見を頂戴しました國枝義敏教授(立命館大学)，上原哲太郎助教授(京都大学)に感謝致します．また，本研究の一部は，文部科学省科学研究費補助金制度若手研究(B)(課題番号17700035)による助成を受けています．

参考文献

[1] :『「相次ぐWinnyによる情報漏えい，原因の一つは厳しすぎる社内ルール」—ISS』, http://itpro.nikkeibp.co.jp/article/NEWS/20051124/225126/ (2005).

[2] : InfoCage, http://www.sw.nec.co.jp/cced/infocage/.

[3] Loscocco, P. and Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System, *Proc of the FREENIX Track, The 2001 USENIX Annual Technical Conference* (2001).

[4] 箱守聡，横山和俊，乃村能成，谷口秀夫: オープン処理に着目した情報拡散追跡法，情報処理学会研究報告 2005-OS-100, pp. 1--8 (2005).

[5] 古瀬淳，米澤明憲: VITC: 対攻撃性コード生成コンパイラ，日本ソフトウェア科学会第22会大会論文集(2005).

[6] Grunbacher, A.: POSIX Access Control Lists on Linux, *USENIX Annual Technical Confernce* (2003).