L-012

# ネットワークへの情報拡散追跡のためのデータ取得
# Implementation of Logging for Information Tracking on Network

**前田 明彦** † Akihiko Maeta　**高橋 健一** † Kenichi Takahashi　**川村 尚生** † Takao Kawamura　**菅原 一孔** † Kazunori Sugahara

## 1　はじめに

近年，計算機性能の向上や普及により今まで紙媒体で管理していた情報を計算機で管理する機会が増大している．また，情報を電子化することで，一度に大量の情報を持ち出すことも可能となり情報の提供や共有が容易になった．しかし，計算機で管理する情報は機密性の高い情報から機密性が低い情報までさまざまであり，機密性の高い情報は外部に漏れてしまうと企業や個人にとって大きな損失になる．個人情報漏洩のインシデント分析結果 [1] によると個人情報漏洩の原因の約 7 割は,「管理ミス」,「誤操作」が占めている．これらの情報漏洩は外部からの攻撃ではなく組織の内部のミスや誤操作が原因である.「管理ミス」とは，情報の公開・管理ルールの明確化や遵守が不十分であることが原因である.「誤操作」については，メールの送信先の宛先の間違いや操作ボタンの押し間違いなど，計算機利用者の不注意が原因である．計算機利用者の不注意は，計算機操作の確認や制御により防止できると考える．そのため，本研究では，原因のひとつである「誤操作」に論点をおく．悪意のある計算機利用者や計算機利用者の意図した機密情報の持ち出しは対象としない．

「誤操作」を防止するための対策として，正当なアクセス権限を持つユーザによる情報漏洩を防ぐことを目的としたオペレーティングシステムの開発 [2] や情報の出力先である計算機資源へのアクセスを制御するシステム [3] が研究されている．また，機密情報を追跡し，機密情報を送信または USB メモリなどの記憶媒体に書き出す際に警告を出すなどして操作を確認及び制御するシステム [4] も研究されている．

以上の既存研究では，単一計算機を対象としたシステムである．しかし，企業などの組織では複数の人間で情報を共有して作業することが想定でき，作業の効率化を考えると，複数の計算機間で機密情報の拡散を追跡し，持ち出しを制限する必要がある．そのためには，まず複数計算機間での機密情報の拡散を追跡することが必要となる．そこで，本論文では，ネットワークへの情報拡散を追跡するためのデータ取得方法を示す．

## 2　機密情報の拡散

機密情報とは，個人情報などの機密性の高い情報が含まれた情報であり，ファイル単位で扱う．図 1 に機密情報が拡散する過程を示す．情報の拡散には，必ずプロセスが関係している．プロセスが情報を読み込み，読み込んだ情報をファイルへ書き出したり，他のプロセスに伝達することで情報が拡散していく．情報の拡散を追跡するためには，どの情報がどのように拡散したか把握する必要がある．このため，ファイル操作とプロセス間通信 (InterProcess Communication:IPC) が発生した場合に，いつ，どこで，誰が，どのような操作でどのファイルを操作したかをログとして記録する．記録されたログをトレースすることで情報の拡散を追跡する．

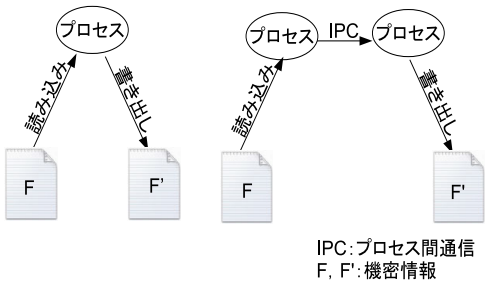


図 1: 機密情報の拡散

また，機密情報の拡散をログとして出力する位置は，オペレーティングシステム (OS) とする．OS でログを出力することで，アプリケーションに依存することなくログを出力できる．ファイル操作では情報の読み書きの際に必ずメモリやハードウェアにアクセスするため，メモリやハードウェアを直接コントロールするカーネルで情報の拡散に関するログを出力することで情報拡散の漏れをなくすことができる．

### 2.1　ファイル操作

ファイル操作による拡散は，機密情報を読み込んだプロセスが，機密情報を他のファイルに書き出すことで発生する．たとえば，あるプロセス P が機密情報 F を読み込み，読み込んだ情報を他のファイル F ’に書き出す．カーネルはプロセスの外部にあるため，プロセスが読み込んだ情報がどのような情報か，プロセス内で読み込んだ情報がどのように参照されたか，編集されたかを把握することは困難である．このため，ファイル操作により機密情報 F がファイル F ’に書き出された可能性があることしか分からない．したがって書き出されたファイル F ’も機密情報として扱う必要がある．

また，ファイルへの操作は，ファイルの読み書き以外にもファイル名の変更やファイルの移動などが行われる．ファイル名の変更では機密情報を作成した時点でのファイル名が変更されるため，誤った持ち出しなどにつながる．たとえば，友人・知人・取引先などの住所を含んだファイルを作成し，ファイル名を ”住所録 ”とした場合は，計算機利用者はファイルの内容は住所が含まれていることが予測できる．しかし，ファイル名の変更によっ

†鳥取大学大学院 工学研究科 情報エレクトロニクス専攻

て ”住所録 ”というファイル名が ”aaa ”などに変更された場合にはファイルの内容に住所が含まれていることは予測することが難しい．したがって，”aaa ”というファイルには住所などの機密性の高い情報が含まれていないと勘違いして持ち出し，持ち出した情報が何らかの原因で外部に漏れて情報漏洩につながる可能性がある．そのため，ファイル名の変更も追跡する必要がある．

ファイルの移動についても機密情報を作成した時点とは別のフォルダに移動することで誤った持ち出しの可能性が生じる．たとえば，機密情報を管理するディレクトリがあった場合，機密情報を違うディレクトリに移動し作業した後に元のディレクトリに戻し忘れると，機密情報を管理するディレクトリ以外に機密情報が存在することとなる．計算機利用者は機密情報が管理されているディレクトリ以外には機密情報は存在しないと思い込み，誤って機密情報の持ち出してしまうかもしれない．そのため，ファイルの移動についても機密情報の拡散を追跡する必要がある．

### 2.2 プロセス間通信 (IPC)

プロセス間通信は，通信機構，シグナル，同期機構の３つの機構に分類できる．その中でプロセス間でのデータのやり取りは，通信機構で発生する．通信機構はさらにデータ転送と共有メモリの２つに分類できる．プロセス間通信による拡散は，機密情報を読み込んだプロセスが，プロセス間通信により他のプロセスに情報を伝搬することで発生する．しかし，2.1 節でも述べたが，カーネルはプロセスの外部にあるため，プロセスが読み込んだ情報がどのように編集されたか把握することは困難である．

## 3 機密情報の拡散追跡

ネットワークへの機密情報の拡散追跡を実現するためには以下のような課題がある．

[課題 1] 機密情報と通常ファイルの区別

[課題 2] 書き出す情報が機密情報であるかの判断

[課題 3] 受信した情報が機密情報であるかの判断

まず機密情報であるかどうかの判断では，プロセスは設定ファイルなどさまざまなファイルを読み込んでいるため，すべての情報の拡散に対してログを出力すると膨大な数となる．設定ファイルなどは，アプリケーションの設定などが書かれているため，機密情報が含まれている可能性は低い．そこで，設定ファイルなどは機密情報とみなさず追跡はしないことが望ましい．そのため，機密情報が含まれているファイルをユーザが指定することで設定ファイルや通常ファイルと区別する．

２つめの課題は，ファイルやプロセスに情報を書き出す際に，書き出す情報が機密情報であった場合，書き出されたファイルやプロセスに機密情報が拡散している可能性がある．このため書き出されたファイルやプロセスを機密情報として扱う必要がある．しかし，カーネルはプロセスの外部にあるのでプロセスが機密情報を書き出したかどうかはカーネルからでは分からない．そのため，プロセスに注目し，情報を書き出すプロセスが過去に機密情報を読み込んでいるかどうかを判断する．情報を書き出すプロセスが過去に機密情報を読み込んでいる場合は，書き出された情報も機密情報である可能性がある．そのため，書き出した情報も機密情報として扱う．

３つ目の課題は，プロセス間通信の際に受信した情報が機密情報であるかの判断である．受信した情報が機密情報である場合は，受信したプロセスにより書き出されたファイルも機密情報として扱わなければ情報漏洩につながる．受信した情報が機密情報であるか判断するには，必ず送信側からの通知が必要となる．この通知をどのように実現するかが複数計算機間で機密情報を送受信する上で重要な課題となる．

## 4 機密情報の拡散を把握するためのデータ取得の実装

linux カーネル 2.6.22 上で実装を行った．拡散の対象は，ファイル操作と子プロセスの生成およびプロセス間通信の中のソケット通信を対象とした．

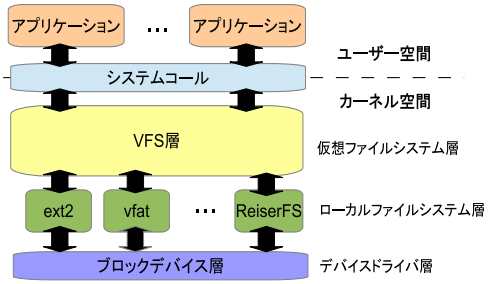


図 2: ファイルシステムの構造

図 2 にカーネル内のファイルシステムの構造を示す．カーネル内でいつ，どこで，誰が，どのような操作でどのファイルを操作したかをログとして出力することでアプリケーションに依存することなくログを出力できる．ユーザーモードに存在するアプリケーションからの命令はシステムコールを通してカーネル内での命令へ変換される．カーネル内では仮想ファイルシステム (VFS 層) でそれぞれのファイルシステムを統一して利用でき，各ファイルシステムの特有の処理はローカルファイルシステムで行われる．しかし，ローカルファイルシステムでログ出力を行う場合，それぞれのファイルシステムにログの出力機能を実装する必要があり，実装に修正があった場合などには，修正漏れなどが発生する可能性がある．そのため，ローカルシステムへ依存することなく統一的にログを出力できる VFS 層でログの出力を行う．

機密情報の拡散を追跡を実現するには，読み込んだファイルが機密情報であるか，書き出す情報が機密情報の可能性があるか，プロセス間通信において受信した情報が機密情報であるか判断する必要がある．実装したシステムの全体図を図 3 に示す．

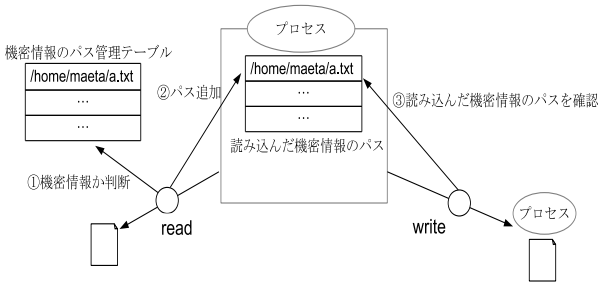


図 3: 実装したシステム

### 4.1 機密情報の判断

ファイル操作による拡散は，情報を読み込んだプロセスが，他のファイルに情報を書き出すことで情報の拡散が発生する．このため，ファイルを読み込んだ際に，読み込んだファイルが機密情報であるか判断する必要がある．そこで，機密情報と通常ファイルを区別するために，カーネル内にユーザ ID と機密情報の絶対パスを組にして管理するテーブルを用意する．ユーザは機密情報として扱いたいファイルを作成した場合，作成したファイルの絶対パスとユーザ ID を登録する．しかし，このテーブルはカーネル内に存在するため，一般的にはカーネルからでなければアクセスできない．そこで，ユーザモードとカーネルモードで情報をやり取りする方法が必要となる．そのため，カーネル内の情報を表示したりカーネル内のパラメータ値の表示や変更が可能である proc ファイルシステムを利用する．proc ファイルシステムを利用することでユーザモードからカーネル内にあるテーブルの機密情報のパスを編集できる．また，ファイルを読み込む関数である vfs_read 関数，vfs_readv 関数で読み込んだファイルのパスとユーザ ID を取得し，それがテーブルに含まれていれば機密情報であると判断できる．

### 4.2 機密情報の読み書きとプロセス間通信でのデータ送信

機密情報はプロセスによって読み込まれ，それが他のファイルやプロセスに書き出されることで拡散する．書き出した情報が機密情報であった場合，書き出した先のファイルも機密情報として扱わなければ，情報漏洩につながる可能性がある．そこで，書き出す情報が機密情報であるか判断する必要がある．情報を書き出しは，write システムコール，writev システムコール，プロセス間通信では send システムコール，sendto システムコールにより，プロセスがファイルや他のプロセスに情報を書き出す．しかし，これらのシステムコールでは書き出す情報はすでにファイルから読み込まれて格納されているため，格納されている情報が機密情報であるか判断することができない．この問題を解決する方法としてプロセスに注目した．

プロセスがファイルを読み込んだ際に機密情報であれば機密情報のパスをすべて保持させておく．プロセスが情報を書き出した場合に，プロセスがパスを保持していれば書き出した情報が機密情報を含む可能性がある．プロセスに機密情報のパスを保持させる実装を示す．Linux は実行する基本的な単位をプロセスとして管理しており，プロセスに関連するさまざまな情報を task_struct という構造体 (プロセスディスクリプタ) に格納している．プロセスディスクリプタは 1 つのプロセスに必ず 1 つ存在し，プロセスとプロセスディスクリプタは 1 対 1 で対応している．そのため，プロセスディスクリプタに機密情報のパスを管理するリストへのポインタを格納するメンバを追加する．read，readv システムコールで機密情報を読み込む毎にこのリストへファイルのパスを追加する．プロセスは情報を書き出すときにこの機密情報のパスを参照し，リストにパスがひとつでも格納されていれば，書き出された情報は機密情報の可能性がある．このため，情報を書き出した際にログを出力し，書き出した先のファイルも機密情報として扱う．したがって，プロセスディスクリプタに読み込んだ機密情報のパスのリストを保持させることで，書き出した情報が機密情報の可能性があるかどうか判断できる．書き出した情報が機密情報の可能性がある場合には 4.1 節で述べたテーブルに追加する．これによって書き出されたファイルも機密情報として扱われ，ファイルの読み書きによる機密情報の追跡が可能である．

### 4.3 プロセス間通信でのデータの受信

プロセス間通信では，プロセス同士で通信しているデータが機密情報であるかどうかはカーネルからは判断することができない．そこで，機密情報を受信した際に受信した情報が機密情報であるかの判断が課題である．この課題はさらに 2 つの課題に分けられる．1 つ目の課題は，受信側では送られてきた情報が機密情報であることを認識することである．そのためには，送信側から受信側への送信した情報が機密情報であることを伝える必要がある．2 つ目の課題は，この通知をどのタイミングで行うかである．

1 つ目の課題の解決策として，送信するデータと一緒に読み込んだファイルのパスのリストを送信する．ファイルのパスのリストを送信することで受信した側でも，送信されたデータが送信元のどのファイルのデータを含む可能性があるか分かる．

2 つ目の課題は通知のタイミングである．受信する前に送信側から通知を受けず，ファイルなどに情報を書き出してしまった場合は機密情報の拡散追跡に漏れが生じてしまう可能性がある．しかし，送信側では受信側が受信したデータを処理するタイミングは分からない．情報を受信してそのままファイルに書き出す場合もあれば，何らかの処理を行ってからファイルやプロセスに書き出すことが考えられる．どちらの場合でも，受信側は情報を受信したときには，受信した情報が機密情報であることを把握しておく必要がある．したがって，送信側では機密情報を送信する前に通知を行うか，機密情報と同時に通知を行う必要がある．本研究では，送信する情報に読み込んだ機密情報のパスを結合させて送信することで

解決する．

## 5　データ取得結果と課題

実際にログを取得した結果を示す．ファイル操作による機密情報の拡散追跡結果について示す．まず，機密情報として/home/maeta/d.txt を登録する．次に，cp コマンドで/home/maeta/d.txt を/home/maeta/e.txt にコピーする．さらに cp コマンドで/home/maeta/e.txt を/home/maeta/k.txt にコピーする．以上の手順を実行した場合のログの出力結果を図 4 に示す．

```
12:26:13 pc1 kernel: File Marked : UID:500 FILE:/home/maeta/d.txt
12:27:22 pc1 kernel: Read File : PID:4108 FILE:/home/maeta/d.txt
12:27:22 pc1 kernel: Write File : PID:4108 FILE:/home/maeta/e.txt DEV:800002
12:27:22 pc1 kernel: File Marked : UID:500 FILE:/home/maeta/e.txt
12:36:57 pc1 kernel: Read File : PID:4312 FILE:/home/maeta/e.txt
12:36:57 pc1 kernel: Write File : PID:4312 FILE:/home/maeta/k.txt DEV:800002
12:36:57 pc1 kernel: File Marked : UID:500 FILE:/home/maeta/k.txt
```

図 4: 機密情報をコピーしたときのログ

1 行目で/home/maeta/d.txt を機密情報として登録され，2 行目で/home/maeta/d.txt を読み込み，3 行目で/home/maeta/e.txt に書き込んだことがログとして出力された．さらに 4 行目で書き込まれたファイル/home/maeta/e.txt は機密情報として登録されたことが確認できる．5〜7 行目で/home/maeta/e.txt を/home/maeta/k.txt にコピーした場合も同様にログが出力され，/home/maeta/k.txt が機密情報として登録されたことが確認できる．この結果より，ファイルの読み書きを追跡し，機密情報が書き出された可能性があるファイルも機密情報として登録できたことが確認できる．

次に，プロセス間通信により拡散したファイルの追跡のためのログの取得結果を示す．プロセス間通信の対象としてはソケット通信を対象とする．そのためファイル (/home/maeta/a.txt) を読み込み，読み込んだデータをサーバに送信するクライアント側のプログラムとクライアントからデータを受信するとファイル (/home/maeta/recv.txt) に書き込むサーバ側のプログラムを作成した．作成したプログラムを実行する前に/home/maeta/a.txt を機密情報として指定し，プログラムを実行した．クライアント側のログの出力結果を図 5(a)，サーバー側のログの出力結果を図 5(b) に示す．

図 5(a) の 1 行目で/home/maeta/a.txt が機密情報として登録され，2 行目で/home/maeta/a.txt を読み込み，3 行目で/home/maeta/a.txt を IP アドレス 127.0.0.1 のポート番号 56303 から IP アドレス 127.0.0.2 のポート番号 10000 に送信していることが確認できる．そしてサーバ側では，図 5(b) の 1 行目では/home/maeta/a.txt の可能性があるデータを IP アドレス 127.0.0.1 のポート番号 56303 から受信していることが確認できる．2，3 行目では受信したデータを/home/maeta/recv.txt に書き込み，機密情報として登録されていることが確認できる．

```
11:12:15 pc1 kernel: File Marked : UID:500 FILE:/home/maeta/a.txt
11:12:29 pc1 kernel: Read File : PID:3751 FILE:/home/maeta/a.txt
11:12:31 pc1 kernel: Send File : PID:3751  send 127.0.0.1:34026  to 127.0.0.2:10000  FILE:/home/maeta/a.txt
```

(a) クライアント側のログ出力結果

```
11:12:31 pc2 kernel: Receive File : PID:3747  receive 127.0.0.2:10000  from 127.0.0.1:34026  FILE:/home/maeta/a.txt
11:12:31 pc2 kernel: Write File : PID:3747 FILE:/home/maeta/recv.txt DEV:800002
11:12:31 pc2 kernel: File Marked : UID:500 FILE:/home/maeta/recv.txt
```

(b) サーバ側のログ出力結果

図 5: ソケット通信によるログ出力結果

## 6　おわりに

本論文では，ネットワークへの情報拡散を追跡するための仕組みを linux カーネルに実装した．これによりファイルの読み書きとソケット通信による他の計算機へのファイル送受信のデータを取得でき，ネットワークへの情報拡散を追跡することが可能となった。

今後の課題として，他のプロセス間通信での機密情報の拡散追跡とログ出力を行うことが課題となる．また，プログラム内で機密情報を読み込んだプロセスがコンソールなどに書き出した場合，コンソールが機密情報を扱うプロセスとしてみなされてしまうため，その後のコンソールでの作業がすべてログとして出力されてしまう．そのため，機密情報をコンソールなどのメモリに書き出した場合の機密情報の扱いを検討する必要がある．

## 謝辞

本研究は科学研究費補助金（23700027，23700098）の支援を受けて行なった．

## 参考文献

[1] NPO 日本ネットワークセキュリティ協会，２０１０年情報セキュリティインシデントに関する調査報告書〜個人情報漏洩編〜，http://www．jnsa.org/result/incident/2010．html

[2] 井田 章三　他，“データフローを主体としたアクセス制御を実現する DF-Salvia の設計と開発”，情報処理学会第 73 回全国大会講演論文集，Vol．2011，No．1，pp．511-513，(3，2011)．

[3] 岩永 真幸，毛利 公一，“4W-2 情報漏洩防止のための出力先毎に制御可能なファイルアクセス制御方式 (認証・アクセス制御，学生セッション，セキュリティ)”，情報処理学会第 71 回全国大会講演論文集，Vol．3，No．1，pp．355 - 356，(3，2009)．

[4] 田端 利宏　他，“機密情報の拡散追跡機能による情報漏えいの防止機構”，情報処理学会論文誌，Vol．50，No．9，pp．2088-2102，(9，2009)．