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

2007－DPS－130 (7)
2007－CSEC－ 36 (7)
2007／3／1

プライバシ情報の伝播範囲を制御するためのOSレベルの通信制御方式

西村 和憲† 鈴来 和久†† 毛利 公一††
† 立命館大学大学院理工学研究科 †† 立命館大学情報理工学部

近年, プライバシ情報の漏洩が問題となっている. その漏洩の原因として, ユーザの誤操作や権限のあるユーザによる悪意ある操作からプライバシ情報がネットワーク上に漏洩する場合がある. この問題を解決するためには, 従来のセキュリティ技術に加えて, 情報漏洩を引き起こすデータ送信を禁止するソフトウェア技術が必要である. そこで, プライバシアウェアOS *Salvia* のネットワーク機構では, プライバシ情報の伝播範囲を制御するために, ソケットシステムコールの実行の可否をファイルのデータ保護ポリシに基づいて判定する. ソケットに関するデータ保護ポリシには, 通信先計算機のIPアドレスとネットマスクを記述できる. 本論文では, その方式と実装について述べる.

A Communication Method in Operating System for Limiting Data Distribution Scope

KAZUNORI NISHIMURA† KAZUHISA SUZUKI†† KOICHI MOURI††
†Graduate School of Science and Engineering, Ritsumeikan University
††College of Information Science and Engineering, Ritsumeikan University

Recently, leakage of privacy information becomes a serious problem. As a cause of the leakage, there is a case that privacy information leaks via network by user's mistaken operation or malicious operation by a user who has authority. To solve this problem, a new software technology that prevents the leakage is necessary in addition to existing security technologies. Therefore we developed a communication control mechanism in privacy-aware OS Salvia to check process's eligibility of executing socket systemcalls based on data protection policy. A policy can be defined for each file which contains privacy information. Users can specify data distribution scope using IP address and netmask in a policy. In this paper, its design and its implementation are described.

1 はじめに

近年, 企業から顧客のプライバシ情報が漏洩する事件が多発している. このようなプライバシ情報の漏洩は, プライバシ情報を管理している企業への訴訟問題や, 情報源であるユーザのプライバシ侵害を引き起こす危険性がある. データ漏洩の要因として, 文献[1]では, (1) ユーザの誤操作, (2) ユーザによる外部への持出し, (3) 情報機器の盗難, (4) ソフトウェアの設定不備に分類されるとしている. プライバシ情報を保存したファイル (以下, プライバシファイルと記す) を保護する技術として, 暗号化, 認証, 侵入検知, サンドボックスなどのセキュリティ技術がある. (3) (4) に起因するデータ漏洩は, これらのセキュリティ技術を用いることで防止できる. (1) (2) に起因するデータ漏洩を防ぐ手法として, 我々は, プライバシアウェアOS *Salvia* [2] を開発している.

Salvia は, Linuxカーネルを基に実装しており, プライバシファイルにアクセスしたプロセスに対してアクセス制御を課すことにより, データ漏洩の原因となりうるプロセスの動作を制限する. *Salvia* は, 計算機やユーザの状態・環境 (以下, コンテキストと記す) に適応したデータ保護を実現し, プライバシ情報を作成・所有する者の意図に反するアクセスを禁止する. これを実現するため, ファイルごとにXMLで記述されたデータ保護ポリシを設定可能としている. データ保護ポリシは, データの伝播範囲に応じて, グループ化されたシステムコールの種類, そのシステムコールの実行を制御するための条件, 制御方法を記述したアクセス制御リストの集合として記述される. また, 制御条件は, コンテキストを用いて記述することができる. これにより, 従来のOSでは実現困難であった, ユーザの誤操作や権限のある悪意を持つユーザへのアクセス制御を実現している.

Salvia では, ファイルアクセス, ソケット・パイプ・共有メモリといったプロセス間通信がアクセス制御の対象となる. 特に, 本論文ではソケット通信に着目し, その制御を実現する手法について述べる. 本手法により, プライバシ情報の伝播範囲を制御し, ネットワークを介したプライバシ情報の漏洩を防止する. 本論文で提案する本手法では, 通信先計算機の特定に通信先計算機のIPアドレスを利用する. プライバシファイルの所有者は, ファイルごとに特定のIPアドレス以外には送信を禁止することができ, プライバシファイルの伝播範囲を制限することができる.

2 ネットワークにおける情報漏洩の事例

近年, 情報漏洩事件は依然として発生している [3]. ネットワークにおける情報漏洩の事例として, クライアント端末からの情報漏洩と FTP・Web サーバからの情報漏洩が挙げられる.

クライアント端末からの情報漏洩の具体例には, 企業内で, 顧客情報を取り扱う業務を担当する従業員が, 顧客情報が保存されているファイルを上司に送信する際に, 誤って社外に送信してしまうといった操作ミスが挙げられる.

FTP・Web サーバからの情報漏洩の具体例には, 企業において, プライバシファイルを格納している FTP サーバに正当なアクセス権限を持つユーザが社外からアクセスすることによって, 社外に顧客情報が漏洩することが挙げられる.

3 関連研究

近年多発している情報漏洩事故の発生要因が, 正当なアクセス権限を持つユーザによる不正アクセスや誤操作である事例が多い. このため, 不正アクセスを防ぐための様々なセキュリティ技術が提案されている. 例えば, リファレンスモニタを適用する手法, アクセス制御方式, 暗号化技術が挙げられる. さらに, これらを組み合わせて適用する手法などが提案されている.

SysGuard[4] は, リファレンスモニタの 1 つで, システムコールの前後でアクセス権限の検査を行うソフトウェアである. アクセス権限の検査は, カーネルに組み込まれたガードと呼ばれるモジュールで行う. SysGuard の特徴は, 多くの種類の専用のガードを組み合わせて適用することにより, ガードの実装を単純化してそれ自体の堅牢性を高めることができる点と, ガードによるアクセス制御の適用範囲を柔軟に設定できる点にある. また, SysGuard は完全性を重視したセキュリティモデルに分類される. *Salvia* では, 秘匿性を重視し, 情報漏洩を防ぐ必要のあるファイルにアクセスしたプロセスに関するシステムコールの発行履歴を保持する. *Salvia* は, この履歴に基づき, 上記のような FTP サーバへのアクセスをシステムコールを制御することにより禁止することができる. また, *Salvia* では, プロセスの振舞いを示すシステムコールの履歴に加えて, アクセス要求が発生した際のコンテキストである計算機の位置や時刻に着目し, コンテキストとデータ保護ポリシに基づくアクセス制御を実現している.

Role-Based Access Cotrol や TE は, Security-Enhanced Linux (以下, SELinux と記す)[6] に用いられている. これらは, ユーザやプロセスごとに計算機資源へのアクセス権限が設定可能である. 最小特権の原則に基づき, ユーザやプロセスに対して必要なアクセス権限のみを設定することにより, アプリケーションの実行に必要となるファイルのみアクセス可能とすることができる. しかし, プライバシデータを含むファイルへアクセスが許可されたユーザやプロセスに対して, ソケットへのアクセス権限が与えられている場合, このユーザやプロセスによって, 2 章で述べた具体例に対処できない. また, SELinux は, 通信先となる IP アドレスを指定したアクセス制御を記述するためのポリシの構文を持たない. IP アドレスを指定したアクセス制御は, Linux カーネルに実装されているパケットフィルタ (iptable) を用いて実現されている. ただし, パケットフィルタによる通信の制御は, 通信の目的にかかわらず, フィルタリングルールに一致するパケットを遮断するため, ファイルの伝播範囲を制限する手法としては制約が厳しすぎる. 一方, *Salvia* では, システムコールの引数をコンテキストとして, アクセス制御の可否の判定に利用できるため, 通信先計算機の IP アドレスとネットマスクを組とする通信処理の制御が実現できる. すなわち, ファイルを特定の計算機にのみ送信を許可するといったポリシを実現することができる.

プライバシ情報の漏洩を防止する研究として, MAC と暗号化を組み合わせた情報漏洩防止システム [5] がある. 文献 [5] の提案手法では, プライバシファイルを全て暗号化し, 取扱い資格をもつユーザのみ暗号復号に必要な鍵をシステムにログインする際に生成する. また, 必要最小限のアクセス権を与えることで, 強制アクセス制御を実現している. この提案では, システム管理者により認められたアプリケーションのみ使用可能であり, 機密ファイルが暗号化されているため, 機密ファイルの送信が可能である. そのため, 権限のある悪意を持ったユーザによって, プライバシファイルを送信され, 送信先において, ユーザのミスから悪意のある第 3 者によって閲覧される危険性がある. *Salvia* では, データ漏洩の危険性のある場合, プライバシファイルの送信自体を禁止する. また, *Salvia* では, OS レベルでの制御を実現しているため, すべてのアプリケーションに適用可能である. さらに, 文献 [5] では, ファイルを一般ファイルと機密ファイルの 2 パターンのみに分類される. そのため, ファイルごとに細かい制御条件を設定することはできない. しかし, *Salvia* では, プライバシファイルごとにデータ保護ポリシを設定可能であるため, ファイルごとに細かい制御条件・制御方法を設定することが可能である.

4 ネットワークにおけるデータ保護

4.1 解決すべき課題

ネットワークにおける情報漏洩は, プライバシファイルを送信する計算機と受信する計算機から発生する. プライバシファイルの送信元となる計算機からの漏洩として, (a) ユーザによるプライバシファイル送信の判断ミスや権限のあるユーザによる悪意ある操作, (b) 送信の際のデータ傍受が挙げられる. また, プライバシファイルの受信先となる計算機からの漏洩として, (c) 受信したプライバシファイルを保護する環境の欠如が挙げられる.

これらに起因する情報漏洩を防ぐために, 次の方式によりデータ保護を行う. (a) による漏洩を防ぐために, プロセスの通信状況とプライバシファイルを読み出した履歴を検査し, コンテキストに基づいたアクセス制御を行い, ファイルを作成・所有する者の意図に反映させたプライバシファイルの送信制御を行う. また, 通信先計算機を特定するために, 通信先計算機の IP アドレスとネットマスクに着目する. (b) による情報漏洩を防ぐために, データ送信を行う際に IPSec を用いて通信路の暗号化を行い, データの傍受を防ぐ. (c) による情報漏洩を防ぐために, 受信先計算機において, *Salvia* によるプライバシファイルの保護機能が稼動しているかを検査する必

要がある. また, 送信するプライバシファイルの保護方法が記述されたデータ保護ポリシを受信先計算機に送信する必要がある. そのために, 受信先計算機での *Salvia* の搭載, 改竄の有無の検査を行う. また, 受信先計算機のユーザを特定することで, より柔軟なアクセス制御を実現させるために, ユーザの検査を行う. さらに, 送信元計算機がプライバシファイルを送信する前に, そのプライバシファイルの組となるデータ保護ポリシを送信する. これらプライバシファイル送信機能により, 受信先計算機においてもコンテキストとデータ保護ポリシに基づいたアクセス制御を行う.

そこで, *Salvia* のネットワーク機構では, 以下の機能を実現する.

- 通信先計算機の IP アドレスに基づくソケット通信制御機能
  - コンテキストの取得
  - ソケット通信の実行可否の判断
- 通信先計算機へのプライバシファイル送信機能
  - 通信先計算機の *Salvia* 搭載, 改竄の検査
  - 通信先計算機のユーザの識別
  - データ保護ポリシの継承

本論文では, 特に, 通信先計算機の IP アドレスに基づくソケット通信制御機能とその実装について述べる. 通信先計算機へのプライバシファイル送信機能に関しては, 今後の課題とする.

4.2 通信先計算機の IP アドレスに基づくソケット通信制御手法

Salvia では, ファイルを作成・所有する者の意図を反映させたファイルへのアクセス制限を行うため, ファイルをオープンしたプロセスに対して, コンテキストとデータ保護ポリシに基づいたアクセス制御を行う. ネットワーク機構では, 通信先計算機を特定するために IP アドレスに着目する.

2 章で述べたクライアント端末からの情報漏洩のケースを防止するためには, プライバシファイルのデータ保護ポリシに上司が使用する計算機の IP アドレスとネットマスクを記述することで, 上司の計算機以外へのデータ送信を禁止する. また, 2 章で述べた FTP/Web サーバからの情報漏洩のケースを防止するためには, 機密ファイルのデータ保護ポリシに社内の IP アドレスのみの記述を行うことで, 社外に情報が漏洩することを防止できる.

このようなデータ保護を実現するために, *Salvia* がプロセスに対して課すアクセス制御の手順を以下に述べる.

1. プロセスが起動される.
2. プロセスによりプライバシファイルがオープンされる.
3. オープンされたプライバシファイルと組となるデータ保護ポリシを *Salvia* のアクセス制御リストに登録する.
4. connect システムコールが発行される.
5. システムコールの履歴と引数を記録する.
6. コンテキストを取得する.
7. アクセス制御リストを参照し, コンテキストに基づいてシステムコールの実行の可否を判断する.
8. 許可された場合, connect システムコールを実行し, 禁止された場合, connect システムコールを失敗させる.

プライバシファイルをオープンする前に connect システムコールが発行された場合, connect システムコールではシステムコールの実行の可否を判断できない. このため, send システムコールでも同様の制御を行う.

5 実　　装

5.1 *Salvia* におけるシステムコールの制御

プロセスは, ファイルをオープンすることにより, ファイルへのアクセスを開始する. そのため, *Salvia* は, プライバシファイルのオープンをトリガとして, プライバシファイルをオープンしたプロセスをアクセス制御の対象とする. *Salvia* は, この制御対象であるプロセスに対し, ファイル, ソケット, パイプなどの計算機資源に対するアクセスを制御する. *Salvia* は, 以下の手順により, これらの計算機資源へのアクセスを実現するシステムコールを制御する.

(1) 制御対象であるか否かにかかわらず, プロセスからのファイル, ソケット, パイプに対するアクセス要求が発生したときのコンテキストを取得・記録する.
(2) プライバシファイルがオープンされたとき, そのデータ保護ポリシを読み出す.
(3) データ漏洩の可能性のあるアクセスが発生したとき, コンテキストと保護ポリシを比較し, そのアクセスの可否を決定する.

以上を実現する, *Salvia* のデータ保護機構の構成を図 1 に示す. *Salvia* では, プライバシファイルと保護ポリシが記述されているファイルは, 対となって保存されている. 保護ポリシに記述されるコンテキストとなるパラメータを収集するのが, History Logger である. History Logger は, プロセスごとにデータをコピーするシステムコールやコンテキストを切り替えるシステムコールの履歴を取得する機能を持つ. History Repository では, History Logger が取得したシステムコールの履歴を時系列データとしてプロセス ID をハッシュ値とするハッシュ・チェイン法を用いて管理する. Access Control List (以下, ACL と記す) では, ファイルとして 2 次記憶装置に格納されているデータ保護ポリシをプライバシファイルオープンをトリガとして ACL に読出し, 読み出したデータ保護ポリシをプロセス ID をハッシュ値とするハッシュ・チェイン法を用いて管理する.

さらに, 保護ポリシに従ってデータを保護するのが, Action Controller である. Action Controller は, ACL で管理している保護ポリシと History Repository で管理しているコンテキストに適応してデータを保護する機能を持つ. 具体的には, システムコールをフックすることによって, データをコピーする処理やコンテキストを切り替える処理を制限する.

プロセスがシステムコールを発行した場合, 制御はシステムコールハンドラから History Logger に移行する. 発行したシステムコールがデータを不正にコピーする危険性がある種類のシステムコールの場合, このシステムコールの履歴を History Repository に登録する. そして, 制御は, History

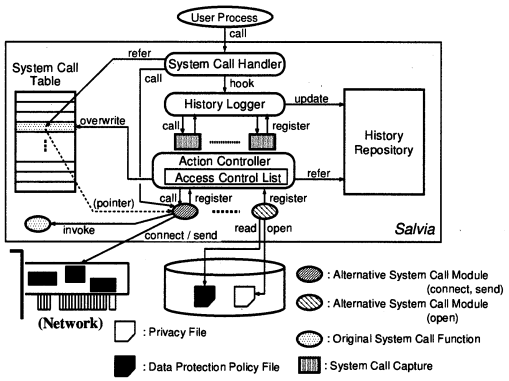

図 1　*Salvia* の全体構成

Logger からシステムコールハンドラ, システムコールハンドラからシステムコールテーブルに登録されている関数に移行する. Action Controller では, このシステムコールサービス関数やこの関数から呼ばれる関数を置き換えることにより, このシステムコールを制御する.

5.2　データ保護ポリシ

Salvia では, データ提供者の意思をデータ保護ポリシとしてファイルごとに定義することにより, データ提供者の意思を反映したプライバシ保護を実現している. データ保護ポリシは, ファイルとして 2 次記憶装置に格納され, 保護ファイルと組にして管理される. データ保護ポリシは, 以下の要素により構成される.

- 制御対象となる操作
- 制御条件
- 制御方法

制御対象となる操作は, 保護ファイルに対してどのような操作を制御するかを示すものであり, 表 1 においてグループとして定義されている要素を用いて記述する. 表 1 は, データ保護ポリシの記述を簡易にするために, データ漏洩を防止する観点から制御する必要のあるシステムコールを分類したものである. おのおののグループは, それぞれ以下の視点から分類した.

- read グループ
  ファイルからデータを読み出すシステムコール群.
- write グループ
  ファイルにデータを書き込むシステムコール群.
- send_local グループ
  同一計算機上の他のプロセスのデータ領域にデータを書き込むシステムコール群.
- send_remote グループ
  他の計算機上のプロセスのデータ領域にデータを書き込むシステムコール群.

制御条件は, 制御を行う条件を示すものであり, コンテキストを用いて記述される. 例えば, 位置や時刻に適応した保護を行う場合, 位置を示すパラメータ値や時刻を記述する. 制御方法は, 制御条件を満たした場合に, 制御対象となる操作に対してどのような制御を行うかを指定するパラメータであり, 「すべて許可」, 「読出し許可」, 「書込み許可」, 「データの読出し以降, 他のファイル・パイプ・ソケットへの書込み禁止」が指定可能である.

5.3　コンテキスト

コンテキストは, ユーザ, 計算機, プロセスについて, それぞれの状況を表す情報である. *Salvia* では, プロセスの属性値, プロセスの動作履歴, 位置, 時刻をコンテキストとして扱う.

- プロセスの属性値
  プロセスの属性値を示すコンテキストとして, 実ユーザ ID, 実効ユーザ ID, プロセス ID を利用する. 実ユーザ ID は, システムコールを発行したプロセスの所有者を示し, 実効ユーザ ID はプロセスがどのユーザの権限で実行されているかを特定するために用いる. 一方, プロセス ID は, *Salvia* においてプロセスごとに履歴を管理するために利用する.
- プロセスの動作履歴
  プロセスの動作履歴を示すコンテキストは, システムコールの履歴により表現される. システムコールの履歴は, システムコール番号, 引数, 返り値で構成される. システムコール番号は, 発行されたシステムコールの種類を特定するために利用し, システムコールの引数は, 引数の内容によりシステムコールを制御するために利用する. システムコールの返り値は, システムコールの結果 (成否やファイルディスクリプタのような識別子) の記録として使用する.
- 計算機の位置
  計算機の位置を示すコンテキストとして, 無線 LAN アクセスポイントの ESSID と電波強度, GPS（Global Positioning System）による測位データを利用する. 無線 LAN を利用する場合, ESSID からアクセスポイントの位置を特定し, 電波強度により計算機とアクセスポイントの距離を推定する. また, GPS を利用する場合は, 緯度と経度を示すパラメータ値から, 計算機の位置を推定する. 位置情報を利用することにより, 計算機の位置に適応した制御が可能となる.
- 時刻
  時刻を示すコンテキストとして, 絶対時刻と相対時刻を利用する. 相対時刻は, ある絶対時刻からの経過時間を示すパラメータであり, 絶対時刻から算出する. 相対時刻を利用することにより, ファイルアクセスが発生した時刻からの経過時間に応じた制御が可能となる.

5.4　通信先計算機の IP アドレスに基づくソケット通信制御機構

ソケット通信を制御するために, System Call Capture と Alternative System Call Module の機能の実装を行った. System Call Capture では, 制御に必要なシステムコールの引数

表 1　データアクセスの分類

分類名	システムコール
read	read, readv, pread64, mmap, mmap2, readahead
write	write, writev, pwrite64, sendfile, sendfile64, mmap, mmap2, munmap
send_local	write, writev, sendfile, sendfile64, mmap, mmap2, socketcall, ipc, mq_timedsend
send_remote	write, writev, sendfile, sendfile64, socketcall

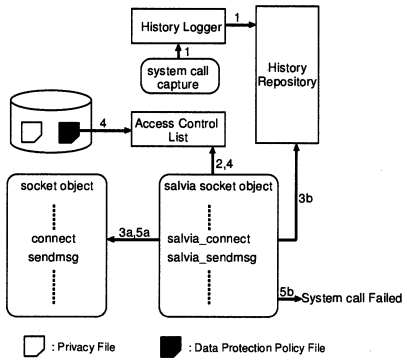

図 2　*Salvia* のネットワーク機構

を履歴として取得する. Alternative System Call Module では, プロセスが socket システムコールを発行した際に, 表 1 の send_remote に分類されたシステムコールが共通して呼び出すソケットオブジェクトの関数ポインタ (connect, sendmsg) を制御を行う関数ポインタ (salvia_connect, salvia_sendmsg) に置き換える. sendfile は, プライバシファイルオープンの際に, ファイルオブジェクトに登録されている関数ポインタ sendfile を salvia_sendfile に置き換える. これらの関数では, コンテキストとデータ保護ポリシに基づいて, システムコールの実行の可否を判断する. 実装を行った各機能の詳細と関数ポインタを置き換えた後の処理の流れは次のとおりである (図 2 参照).

1 connect, sendto システムコールが発行された場合, System Call Capture では, システムコールの引数から通信先計算機の IP アドレスとソケットディスクリプタを History Logger に通知し, ユーザ ID , 時刻と共にプロセス ID ごとに時系列データとして History Repository に登録する. また, accept システムコールが発行された場合は accept システムコールの処理後に, System Call Capture と同様のパラメタを History Repository に登録する.

2 制御の必要のあるシステムコール (connect, send 系) が発行されると, Alternative System Call Module では, 過去に当該プロセスが, プライバシファイルを読み出しているかをプロセス ID をキーとして ACL を検索する.

3.a プライバシファイルが読み出されていない場合, データ漏洩の危険性がないため, 通常のソケット通信の処理を行う.

3.b プライバシファイルが読み出されている場合, データ漏洩の危険性があるため, コンテキストとデータ保護ポリシに基づいたソケット通信制御を行う. そのため, 現在の通信状況を取得するため, ソケットディスクリプタとプロセス ID を識別子として, History Repository から, 通信先計算機の IP アドレスを検索し, 一致すればその通信先計算機の IP アドレスを取得する.

4 プライバシファイルがオープンされた際に, XML で記述したデータ保護ポリシファイルを OS のみ参照できる ACL に登録する. プロセス ID をキーとして, 当該プロセスがオープンしているプライバシファイルのデータ保護ポリシを登録されている ACL から IP アドレスとネットマスクの組を検索し, データ保護ポリシとして取得した IP アドレスとコンテキストとして取得した IP アドレスをデータ保護ポリシとして取得したネットマスクとの論理積をとり, 生成されたデータ保護ポリシの IP アドレスとコンテキストの IP アドレスを比較する.

5.a 一致した場合は, システムコールの実行が許可されるため, 通常のデータ送信を行う Original System Call Function を呼び出す.

5.b 一致しない場合は, システムコールの実行が禁止されるため, データ送信を失敗させる関数を呼び出す.

6 評　　価

6.1 機能評価

2 章で挙げた FTP サーバを介したデータ漏洩の具体例を防ぐために, 社内の計算機が保持する IP アドレスのみ送信を許可し, 社員のみプライバシファイルの読出し・更新を許可する. そのために, データ保護ポリシに図 3 の記述を行う. 社員を示すグループ ID の値を 1001, 社内のネットワークを 192.168.20/24 とする. 図 3 の 4-13 行目の default_access タグとその要素は, 社員以外のユーザがプライバシファイルを読み出す場合と社内の計算機以外にプライバシファイルを送信する場合のアクセス権限を記述する部分である. 14-32 行目の data_protection_domain タグの属性を none にすることで, このポリシがプライバシファイルの読出しとその更新を制御するポリシを記述できる. 17-19 行目の group タグとその要素は, 社員であるユーザに対しての制御であることを記述している部分である. 26-29 行目の send_remote タグとその要素は, 送信を許可する計算機の IP アドレスとネットマスクの組を記述している部分である.

このような記述を行うことで, 表 1 で send_remote に分類されたシステムコールの実行を制御することができる.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>

2 <!DOCTYPE data_protection_policy SYSTEM "policy.dtd">

3 <data_protection_policy>

4  <default_access>
5   <read>deny</read>
6   <write>
7     <write_access updata="deny">deny</write_access>
8   </write>
9   <send_local>deny</send_local>
10   <send_remote>
11     <send_remote_access>deny</send_remote_access>
12   </send_remote>
13 </default_access>

14 <data_protection_domain type="none">
15 <ACL>
16   <context>
17     <group>
18       <group_id type="own>1001</group_id>
19     </group>
20   </context>

21   <access>
22     <read>allow</read>
23     <write>
24       <write_access updata="allow">deny</write_access>
25     </write>
26     < send_remote >
27       < send_remote _access > allow < /send_remote _access >
28         < ip_address > 192.168.20.0/24 < /ip_address >
29     < /send_remote >
30   </access>
31 </ACL>
32 </data_protection_domain>
33 </data_protection_policy>
```

図 3　XML によるデータ保護ポリシの記述例

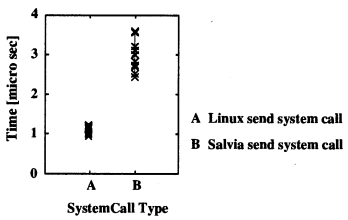

図 4　send システムコールの処理時間

6.2 性能評価

Salvia のネットワーク機構の性能評価のため, 6.1 節で示したデータ保護ポリシに従ったシステムコール処理の制御にかかるオーバヘッドを計測した. 実験環境は, CPU が Intel Pentium III Processor 900 MHz, メモリ 128MB の PC/AT 互換機を用いた.

実験では, 10KB のプライバシファイルを送信する際のカーネル内部の Linux の send システムコールと *Salvia* の send システムコールの処理時間を計測した. 計測方法は, RDTSC (read-time stamp counter) 命令を使ったクロックサイクル数の計測を用いた. *Salvia* の send システムコールでは, Linux の send システムコールの機能に加えて, (a) プライバシファイルの検索, (b) コンテキストの検索, (c) コンテキストと保護ポリシの比較の機能を有する.

Linux の send システムコールの処理時間は, 約 1 μ 秒である. *Salvia* の send システムコールの処理時間は, 約 3 μ 秒であり, Linux の機能を除いた内訳は, (a) 約 0.8 μ 秒, (b) 約 0.5 μ 秒, (c) 約 0.7 μ 秒であった. *Salvia* の send システムコールの処理時間の増加率は, Linux の send システムコールの約 3 倍となった (図 4 参照). 一方, 当該プロセスが

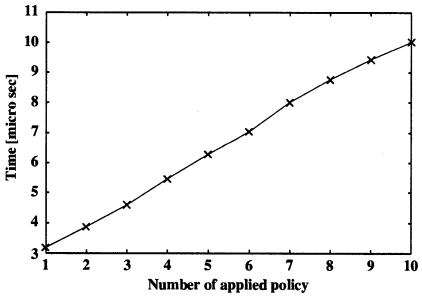

図 5　FTP を利用した際の処理時間

プライバシファイルを読み出していない場合は, プライバシファイル読出しの検査以外は, Linux の send システムコールと同一の処理を行っているため, オーバヘッドは 0.8 μ 秒程度である. また, FTP サーバでは, 1 つのプロセスが複数のファイルにアクセスを行う. そのため, 本手法ではプライバシファイルを読み出す数に応じて, プロセスに課せられる制約が厳しくなる. その際の (c) の処理時間は, 適用されたデータ保護ポリシの数に比例して増加するため, *Salvia* の send システムコールの処理時間も同様に増加する (図 5 参照).

7 おわりに

本論文では, プライバシ情報の伝搬範囲を制御するために, 解決すべき課題を 3 点挙げ, 特に, ソケット通信を制御する手法とその実装について述べた. 本手法では, 通信先計算機を特定するため, IP アドレスに着目し, コンテキストとデータ保護ポリシに基づいて, 表 1 の send_remote に分類されるシステムコールの実行を IP アドレスとネットマスクの組を用いて制御する. 今後の課題として, 残りの 2 点の課題の解決を行う.

参考文献

[1] 独立行政法人国民生活センター: 個人情報流出事故に関する事業者調査結果, http://www.kokusen.go.jp/news/data/n-20050325_1.html

[2] 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.SIG3, pp.1-15(2006).

[3] Security NEXT:個人情報漏洩事件一覧, http://www.security-next.com/cat_cat25.html

[4] 榮樂 恒太郎, 新城 靖, 板野 肯三: システム・コールに対するラッパ/リファレンスモニタ SysGuard の設計と実装, 情報処理学会論文誌, Vol.43, No.6, pp.1690-1701(2002).

[5] 荒井 正人, 甲斐 賢, 永井 康彦, 富田 理情報漏洩防止システムの提案, 研究報告:コンピュータセキュリティ, Vol.2004 No.22, pp.61-67(2004).

[6] National Security Agency:Security-Enhanced Linux, http://www.nsa.gov/selinux/.