1J-3

プライバシアウェア OS *Salvia* における 共有メモリを介したデータ漏洩の防止手法

廣 真文 † 鈴来 和久 †† 毛利 公一 ††† 大久保 英嗣 †††
† 立命館大学理工学部 †† 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

1 はじめに

近年，計算機に保存されているプライバシ情報が漏洩する事故が発生している．例えば，顧客情報や機密情報が企業の計算機から漏洩し，悪用される事件が頻発している．データ漏洩を防止するための技術としてセキュリティ技術が挙げられるが，従来のセキュリティ技術は，第三者による攻撃から計算機を保護することを目的としているため，人為的なミスによるデータ漏洩を防止することは困難である．また，移動端末の普及に伴い，計算機を取り巻く環境が頻繁に変化する状況が想定されるため，このような状況においてもプライバシ情報の保護が可能な技術を開発する必要がある．

以上の背景より，現在，我々は，ユーザと計算機の状況（以下，コンテキスト）に適応したプライバシ保護を実現するオペレーティングシステム（以下，OS）*Salvia*[1] を開発している．*Salvia* は，保護を必要とするファイル（以下，保護ファイル）ごとにデータ保護ポリシを定義し，保護ファイルをオープンしたプロセスに対して，データ保護ポリシとコンテキストに基づくアクセス制御を実現する OS である．*Salvia* においては，ファイルアクセス，ソケット通信，パイプや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．

本稿では，特に，*Salvia* において，共有メモリを用いたプロセス間通信によるデータ漏洩を防止する手法を提案する．本稿で提案する手法は，共有メモリに起因するデータ漏洩を防ぐために，共有メモリに対する書込みアクセスを制御する．これにより，複数のプロセスがプライバシ情報を共有することを禁止する．

2 共有メモリを介したデータ漏洩の防止

2.1 データ漏洩が発生するシナリオ

プロセスは，System V IPC で提供されている共有メモリ資源を利用することにより，複数のプロセス間でデータを共有することができる．共有メモリを介したデータ漏洩の例を以下に示す．

Apache のようなウェブサーバを始めとして，プロセス間で大量のデータを共有するアプリケーションでは，共有メモリによるプロセス間通信が頻繁に利用される．これらのアプリケーションのように，複数のプロセスが共有メモリによりデータを共有する状況を想定する．例えば，2 つのプロセス（プロセス A とプロセス B ）が，同一の共有メモリをアタッチする．ここで，プロセス A が保護ファイルをオープンし，共有メモリに保護ファイルの内容（以下，保護データ）を書き込むと，プロセス B は保護データを共有メモリから読み出すことができる．このように，複数のプロセスが保護データを共有することにより，データが漏洩する危険性がある．

2.2 共有メモリのアクセス制御モデル

メモリアクセスは，システムコールを介さずに CPU のインストラクションにより実行されるため，カーネルはこれを把握することができない．そのため，共有メモリへの書込みアクセスを制御するためには，メモリアクセスをフックし，カーネルに処理を遷移させる必要がある．また，共有メモリのアタッチと保護ファイルのオープンの実行順序に関係なく，両者が実行された時点からアクセス制御を開始する必要がある．

そこで，共有メモリへの書込みアクセスをフックするために，ページフォルト例外を利用する．ページングによるメモリ管理の場合，ページフォルト例外を発生させることによりハンドラに処理が遷移するため，これをトリガとしてアクセス制御を実現する．共有メモリをアタッチするシステムコール shmat が発行されたとき，カーネルは共有メモリリージョンの情報を取得する．プロセスが保護ファイルをオープンした後に共有メモリをアタッチした場合，ページフォルトが発生するように取得した共有メモリの情報を基にページテーブルを変更する．また，プロセスが共有メモリをアタッチした後に保護ファイルをオープンした場合，保護ファイルのオープンをトリガとして，取得済みの共有メモリの情報を基にページテーブルを変更する．

これにより，共有メモリへの書込みアクセスに対してページフォルト例外を発生させることができる．ハンドラ内でコンテキストと保護ポリシを比較することにより，コンテキストに適応した共有メモリへの書込みアクセスの制御を実現する．

2.3 データ保護ポリシ

前節で述べたアクセス制御モデルを実現するために，データ保護ポリシに共有メモリへの書込みの可否を記述

A Preventive Method of Data Leakage via Shared Memory in *Salvia*

Masafumi Hiro†, Kazuhisa Suzuki††, Koichi Mouri†††, and Eiji Okubo†††
†College of Science and Engineering, Ritsumeikan Univ.
††Graduate School of Science and Engineering, Ritsumeikan Univ.
†††College of Information Science and Engineering, Ritsumeikan Univ.

```
<Control Condition> : read : allow
<Control Condition> : send_local : deny
```

図 1　データ保護ポリシの記述例

する．*Salvia* では，データ保護ポリシの記述を容易にするために，データ漏洩を防止する観点からシステムコールを分類し，それぞれのグループについてポリシを記述する [1]．共有メモリアクセスに起因するデータ漏洩を防ぐポリシの例を図 1 に示す．このポリシは，ファイルの読出しを許可すること，共有メモリを含む同一計算機内のプロセスに対する通信を禁止することを示している．また，時刻や位置といったコンテキストを用いて制御条件を記述することで，その条件下でのみ前述の制御を可能とする．

3　共有メモリアクセス制御によるデータ保護

3.1　*Salvia* の概要

Salvia は，Linux カーネルを基に Intel x86 アーキテクチャ上に実装している（図 2 参照）．*Salvia* では，制御対象となるシステムコールについて，システムコールテーブルに登録されている関数ポインタをシステムコールの実行を制御するモジュール（Alternative System Call Module）を指すように置き換える．History Logger は，システムコールの履歴を取得する．取得した履歴は，プロセスごとに History Repository に格納される．Action Controller は，History Repository に格納されている履歴と，時刻や位置といったコンテキストを参照し，データ保護ポリシと比較して実行の可否を決定する．Alternative System Call Module は，実行の可否に基づき，システムコールを制御する．

3.2　共有メモリの書込みアクセス制御

共有メモリのアクセスの制御手順を以下に示す．

1. ipc システムコールが発行され，do_shmat が実行された場合，共有メモリリージョンの先頭リニアアドレスを do_shmat の返り値から取得する．

2. do_shmat の実行が終了すると，History Repository を検索し，保護ファイルがオープンされているか確認する．保護ファイルがすでにオープンされている場合は，共有メモリリージョンに対応するページテーブルの書込みビットをクリアする．一方，保護ファイルがオープンされていない場合は，取得したリニアアドレスを履歴として保持し，これ以降に保護ファイルがオープンされた場合に，対応するページテーブルの書込みビットをクリアする．

3. 保護ファイルをオープンしたプロセスが共有メモリに書込みを行うと，ページフォルト例外が発生し，例外ハンドラに処理が遷移する．ハンドラは，引数のエラーコードとメモリリージョンの情報からフォルトの発生原因を特定する．

4. フォルトの原因が共有メモリへの書込みアクセスの場合，コンテキストを取得し，データ保護ポリシと比較して共有メモリへの書込みの可否を決定する．

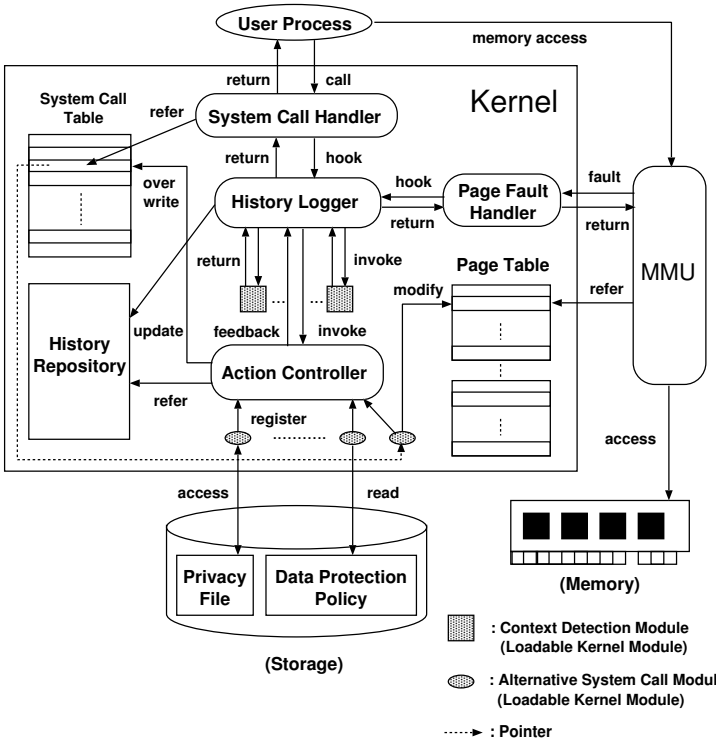

図 2　*Salvia* の構成

本手法により，保護ファイルのオープンと共有メモリのアタッチの順序にかかわらず，共有メモリへの書込みアクセスを制御することができる．また，共有メモリへ書込みアクセスが発生するたびにコンテキストを取得し，それに基づいた制御を行うため，コンテキストに適応した保護が実現できる．

4　おわりに

本稿では，プライバシアウェア OS *Salvia* における，共有メモリを介したデータ漏洩を防止する手法について述べた．本手法により，コンテキストに適応した共有メモリアクセスの制御が可能となる．今後の予定として，データ保護ポリシをプロセス間で継承することにより，より柔軟な共有メモリアクセスの制御手法を検討する．

参考文献

[1] 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（ACS 13），掲載予定．