コンパイラとOSの連携によるデータフロー間伝播解析

樫山　武浩†　　瀧本　栄二†　　毛利　公一†

†立命館大学
525-8577 滋賀県草津市野路東 1-1-1
{kashiyama, takimoto, mouri}@asl.cs.ritsumei.ac.jp

あらまし 情報漏洩を防止するために，データフローを主体としたアクセス制御を実現するセキュアOS「*DF-Salvia*」の開発を行っている．これまで，*DF-Salvia*では，コンパイラにおいて静的解析したデータフロー情報に基づいて，プロセス内における保護データの使用を監視してきた．本稿では，変数間の代入や関数呼出しによって発生するデータフロー間のデータ伝播を動的に解析する手法を提案する．本手法では，実行時情報をもとに，データフロー間の結合状態を動的に決定することにより，変数間や関数を跨ぐ広域なデータフローを細粒度に解析することが可能となる．

Analysis of Data Spread between Data Flows in Cooperation with Compiler and OS

Takehiro Kashiyama†　　Eiji Takimoto†　　Koichi Mouri†

†Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577 JAPAN
{kashiyama, takimoto, mouri}@asl.cs.ritsumei.ac.jp

Abstract We have been developing operating system *DF-Salvia* which provides mandatory access control based on data flow for the purpose of preventing data leakage. *DF-Salvia* observes behavior of processes based on data flow information analyzed in compiler. In this paper, method for analyzing data spread between data flows dynamically is described. The proposed method decides bonding state between data flows according to the running state. As a result, it becomes possible for wide-area data flow that steps over variable and functions to be analyzed.

1　はじめに

近年，社会問題化している情報漏洩の多くは，管理ミス，誤操作，盗難・紛失といった正当なアクセス権限を持つユーザの人為的ミスを要因としている [1]．これらの情報漏洩は，暗号化や認証などの外部からの攻撃を防ぐことを目的としたセキュリティ技術で防ぐことは難しい．また，WindowsやSELinux等に実装されるアクセス制御機構では，操作主体（プロセス）から操作対象（データ）に対するアクセス自体を制限するため，情報漏洩防止の目的からすると，データの機密性を意識されておらず，制限が厳しくなりすぎ利便性が損なわれる．

以上の背景から，操作主体がどのように操作対象を使用するかに応じたアクセス制御を実現することが，情報漏洩の防止・利便性の確保の観点において望まれている．具体的には，「データの読込みを許可するが，それを外部に出力することは禁止する」といったアクセス制御を実現する必要がある．また，アクセス制御の方針

は，操作対象ごとに任意に設定できる必要がある．さらに，社会への導入を考慮すると，OS 内にアクセス制御機構を有し，透過的に全てのアプリケーションに適用できることが望ましい．

上記の要求を満たすアクセス制御には，OS 内において，プロセスがデータを読み込んだのち，そのデータをどのように使用するか，つまりプロセス内のデータフローを把握する仕組みが必要である．これまで，静的解析したデータフロー情報に基づくことで上記の要求を満たすアクセス制御を実現する *DF-Salvia*[2] を開発してきた．*DF-Salvia* では，プロセスの実行前にデータフロー解析が完了しているため，アクセス制御のオーバヘッドが少なく，ユーザに与える負担は小さい．しかし，*DF-Salvia* では，これまで変数単位に自動生成した局所的なデータフロー情報のみを対象としている [3]．そのため，プロセス内のデータの流れを完全に把握できず，情報漏洩を防止できない場合がある．

そこで，*DF-Salvia* への適用を目的に，プロセス内における広域なデータフローを解析する手法を提案する．本手法では，局所的なデータフローを静的解析することで実行時のオーバヘッドを抑制し，かつ静的解析では過剰なアクセス制御を引き起こす要因となるデータフロー間のデータ伝播を動的解析することで高いユーザビリティの確保を実現する．

2 *DF-Salvia*

2.1 概要

DF-Salvia は，情報漏洩の防止を目的としたアクセス制御機構を備えるセキュア OS である．*DF-Salvia* では，保護対象のファイル (以下，保護ファイル) ごとに，データ提供者の意図する保護方針をデータ保護ポリシ (以下，ポリシ) として管理する．ポリシでは，ユーザ ID，時刻，計算機の位置，送信先計算機の IP アドレスなどのプロセスや計算機の状況を示すコンテキストを使用することで,「保護データへのアクセスは許可するが，USB メモリへのコピーは禁止する」,「ネットワーク上への送信は禁止する」といった設定を可能としている．そして，プロセスの動作を監視し，保護データの格納元である保護ファイルに設定されたポリシに従って，プロセスによる保護データの外部への書き出しを制御することを可能とする．

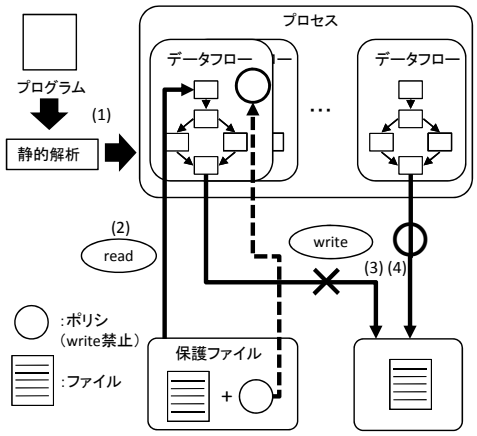

図 1: アクセス制御モデル

2.2 アクセス制御モデル

DF-Salvia におけるアクセス制御モデルを図 1 に示す．DF-Salvia では，プログラムから静的解析したデータフロー情報に基づいて，以下の手順によりアクセス制御を行う．

1. プロセスの実行前に静的解析したデータフロー情報を受け取る．
2. read システムコールが発行された時，読み込み対象が保護ファイルなら，読み込まれるデータが属するデータフローに対して，対象保護ファイルのポリシを適用する．
3. write システムコールが発行された時，書き出されるデータが属するデータフローに対して適用されたポリシの有無を確認する．
4. ポリシが適用されている場合は，そのポリシに従ってシステムコールの実行の可否を判断する．

上記の処理を行うためには，手順 2，3 において，システムコールが発行された時点で使用されているデータが属するデータフローを OS

が特定できなければならない．データフローは，システムコールの発行元であるライブラリ関数コールの命令アドレスから求める．事前にプログラム解析するデータフロー情報として，read システムコールを発行するライブラリ関数コールを定義点，write システムコールを発行するライブラリ関数コールを使用点とする定義-使用連鎖 [4] を使用する．また，それぞれの定義点，使用点について，ライブラリ関数コールの命令アドレスを求めておく．そのため，システムコールにおいて取得した命令アドレスとデータフロー情報に含まれる命令アドレスを比較することで，データフローを特定できる．なお，OS 実行時のライブラリ関数コールの命令アドレスは，システムコール発行時にプロセスのスタックを解析することで求める．

3　データフロー間伝播解析

DF-Salvia がデータフロー情報として使用する定義-使用連鎖は，変数ごとに表現されるデータフローである．そのため，*DF-Salvia* では，変数や関数間を跨ぐ広域なデータフローを把握できず，情報漏洩を完全に防止できない．そこで，本稿では，データフロー間におけるデータ伝播を解析する手法を提案する．

広域なデータフローは，プログラム中に代入文や関数コールを契機として，個々のデータフローを結合することで静的解析できる．その場合，*DF-Salvia* のアクセス制御において，1 つのデータフローに複数の定義点が属することになり，データフローに対して同時に複数のポリシが適用される場合がある [5]．この場合，write システムコールにおいて，発行元の使用点が属するデータフローを特定したのち，さらにデータフローに適用されたポリシの内，どのポリシをアクセス制御に課すべきかを判別する必要がある．しかし，*DF-Salvia* では，ポリシを判別できず，データフローに適用された全てのポリシを使用点でのアクセス制御に透過的に課すため，過剰なアクセス制御が発生する．なお，ポリシ決定の問題については，文献 [5] において詳細を述べている．

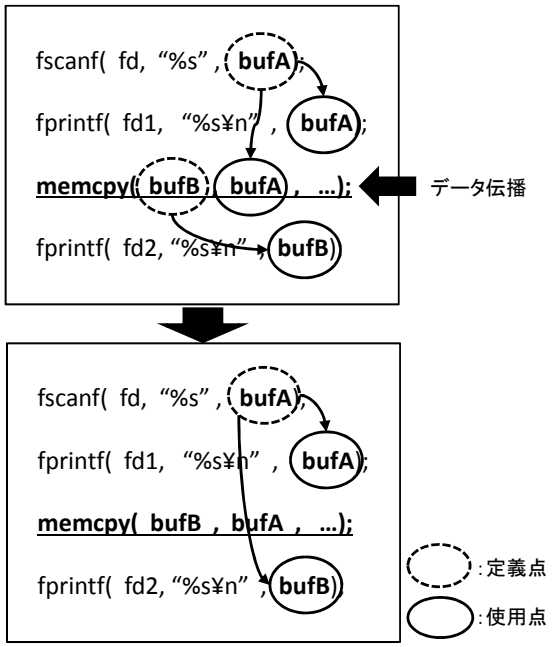

図 2: データフローの結合

そこで，提案手法では，データフロー間のデータ伝播を実行時情報に基づいて動的解析する．これにより，データフローに同時に複数のポリシを適用する機会をなくし，write システムコールにおいて，データフローから一意にポリシを決定することを可能とする．つまり，アクセス制御におけるポリシ決定の問題を解決する．

ただし，変数により，静的解析時にデータフローを結合することにより，アクセス制御時にポリシ決定の問題が発生しない場合がある．この場合，静的解析によりデータフローを結合することで，動的解析を行う機会をなくし，アクセス制御のオーバヘッドの発生を抑制する．

4　静的なデータフロー結合

変数間のデータ伝播において，代入文や関数コール等のデータ伝播を発生させる文以降に，伝播先と伝播元の変数が再定義されない場合，それらの変数のデータフローを結合する．

データフローの結合例を図 2 に示す．図 2 では，変数 bufA と変数 bufB のデータフローが存在し，ライブラリ関数 memcpy によって変数

bufA から変数 bufB へのデータ伝播が発生する．また，ライブラリ関数 memcpy 以降において，変数 bufA と bufB は再定義されない．このようなプログラムでは，2 つのデータフローを結合したとしても，結合後のデータフローの全ての使用点において，変数 bufA の定義点で読み込まれたデータのみが到達する．つまり，データフローに複数のポリシが適用されることがなく，使用点でのアクセス制御において，一意にポリシを決定できる．そこで，図 2 下に示すように，静的解析時において，変数 A と変数 B のデータフローを結合する．

また，定義-使用連鎖では，1 つの変数に複数の定義点が存在するとき，それぞれ別のデータフローとして静的解析される．しかし，それらのデータフローには，一方のデータフローにポリシが適用された場合，もう一方のデータフローに属する使用点でのアクセス制御にもそのポリシを適用しても問題はない．なぜなら，1 つの変数の複数のデータフローに同時に異なるデータが流れないためである．そこで，変数に複数のデータフローが存在する場合は，それらを 1 つのデータフローとすることで，管理すべきデータフロー数を削減する．

5 動的なデータ伝播解析

データフロー間のデータ伝播を動的解析するためには，実行時に OS が以下の事項を把握できる必要がある．

- データ伝播のタイミング
- データ伝播元と伝播先のデータフロー

前者については，静的解析時において，データフロー間のデータ伝播を OS に通知するシステムコール（以下，代入通知システムコール）をプログラムに挿入することで OS に通知する．後者については，挿入した代入通知システムコールを，伝播元のデータフローの使用点として，代入先のデータフローの定義点としてデータフロー情報に含めることで OS に通知する．そして，代入通知システムコールの実行時に，以下

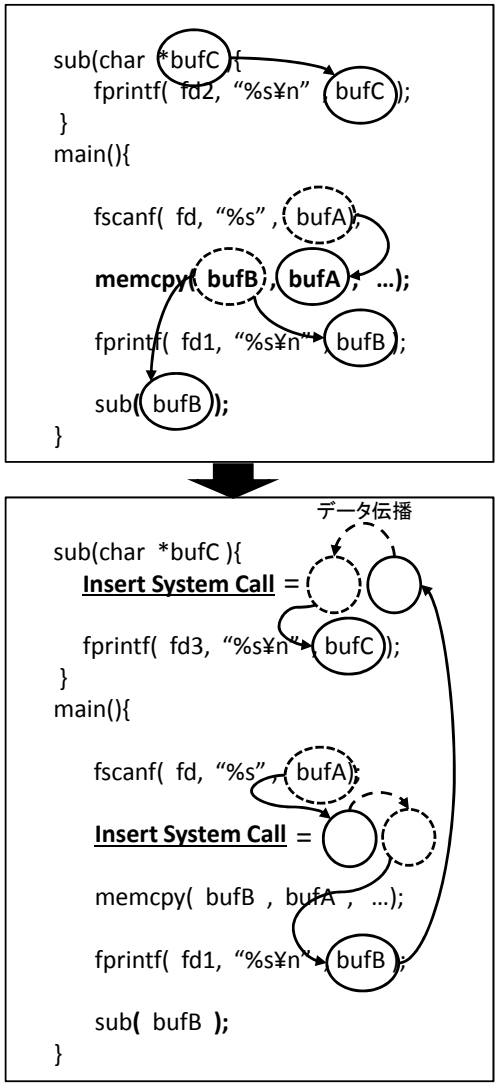

図 3: 代入通知システムコールの挿入

の手順によって，データフロー間のデータ伝播を動的解析する．

1. 代入通知システムコールが使用点として属するデータフローを特定する．
2. 代入通知システムコールが定義点として属するデータフローを特定する．
3. 手順 1 のデータフローに適用されるポリシを手順 2 のデータフローに伝播させる．

代入通知システムコールの挿入例を図 3 に示す．図 3 では，関数 main 内のライブラリ関数 memcpy によって変数 bufA から変数 bufB へ

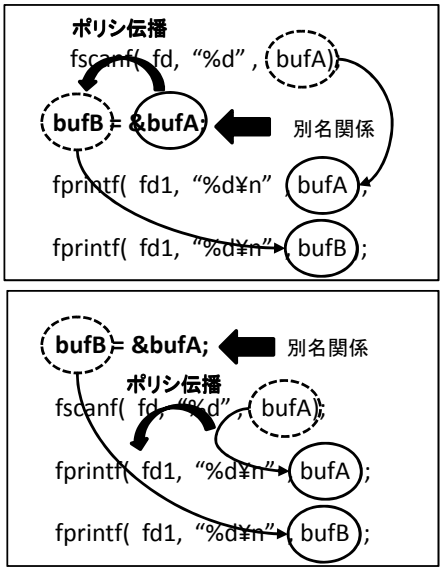

図 4: 別名によるデータ伝播

のデータ伝播が，関数 main 内の関数 sub の関数コールによって変数 bufB から変数 bufC へのデータ伝播が発生する．前者では，関数 memcpy の直前に代入通知システムコールを挿入し，その命令文を bufA のデータフローの使用点として，bufB のデータフローの定義点として登録する．後者では，関数 sub の先頭に代入通知システムコールを挿入し，その命令文を変数 bufB のデータフローの使用点として，変数 bufC のデータフローの定義点として登録する．実行時において，データフロー情報に基づいて，上記で述べた動的解析の手順を行うことで，変数 A から変数 B，そして変数 C のデータフローにポリシを適用する．

6 別名を考慮したデータ伝播解析

ある変数と別名関係を持つ変数のデータフロー間についても，変数間の別名関係を生成する文を，値の伝播を発生させる文と同様に扱い，代入通知システムコールを挿入することでデータフロー間のデータ伝播を動的解析できる．しかし，別名関係の生成時に，データフロー間でポリシを伝播することで，不正なアクセス制御が発生する場合がある．

図 4 のプログラムでは，変数 bufA とその別名を持つ変数 bufB のデータフローがそれぞれ存在する．図 4 上のプログラムでは，別名関係を生成する文を契機として，代入通知システムコールを挿入することで，変数 bufA の定義点で読み込まれたデータのポリシを変数 bufB のデータフローにも適用できる．一方，図 4 下のプログラムでは，別名関係が生成された後に実行される変数 bufA の定義点が，変数 bufA から変数 bufB のデータフローにポリシを伝播すべき地点となる．この場合，前述した代入通知システムコールによる手法では，アクセス制御に課すべきポリシを変数 bufB のデータフローに適用できない．

そこで，変数の別名関係を OS 内で管理することで，別名関係を持つ変数におけるデータフロー間のデータ伝播を動的解析する．具体的には，プログラム中の別名関係を生成する文を契機として，システムコール（以下，別名関係通知システムコール）を挿入し，別名関係を持つそれぞれの変数のデータフローに使用点として登録する．そして，別名関係通知システムコールの実行時に，以下の手順により，OS 内でデータフロー間のポリシ伝播関係を管理する．なお，ポリシ伝播関係とは，データフローにポリシが適用された際，それを伝播させるデータフローへの参照情報を示す．

1. 別名関係通知システムコールが使用点として属するデータフロー群を特定する．
2. 手順 1 で特定したデータフロー間のポリシ伝播関係を OS 内に登録する．

そして，通常の read システムコールと代入通知システムコールでのアクセス制御に，以下の手順を加えることで，データフロー間のデータ伝播を動的解析する．

1. システムコールの発行元が属するデータフローを特定する．
2. 1 で特定したデータフローにポリシ伝播関係が存在する場合，参照先のデータフローにポリシを伝播させる．

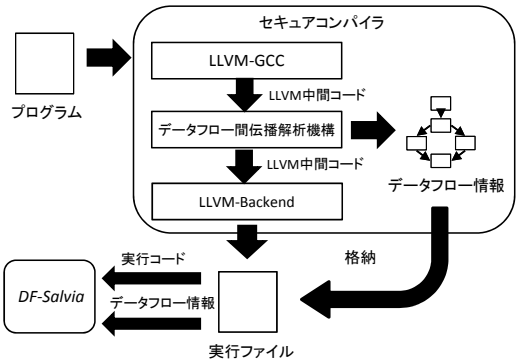

図 5: セキュアコンパイラの構成

上記の手順 2 は，参照先のデータフローにおいて再帰的に実行し，変数を跨ぐ参照先のデータフローにもポリシを伝播させる．

7 実装方法

コンパイラインフラストラクチャとして開発される LLVM（Low Level Virtual Machine）を使用して，本稿で述べたデータフロー間伝播解析機構を備えたセキュアコンパイラを実装している．セキュアコンパイラの構成を図 5 に示す．

まず，プログラム言語として C 言語を対象とすることから，コンパイラのフロントエンドとして提供される LLVM-GCC を使用して，対象プログラムから LLVM 中間コードを生成する．そして，LLVM 中間コードにおいて，本稿で述べたデータフロー間伝播解析を行い，データフロー情報の生成と，LLVM 中間コードへの代入通知システムコール・別名関係通知システムコールの挿入を行う．最後に，LLVM のバックエンドを使用して実行ファイルの生成を行う．なお，静的解析したデータフロー情報については，実行ファイル中のセクションに格納することで，*DF-Salvia* に伝達する．

8 おわりに

本稿では，*DF-Salvia* におけるアクセス制御の精度向上を目的としたデータフロー間伝播解析手法について述べた．本手法では，変数ごとに生成されるデータフロー間におけるデータ伝播を動的解析することで，静的解析したデータフローのみに依存した場合に比べ，アクセス制御時に課すべきポリシを正確に決定することが可能となる．また，プロセス内の広域なデータフローを追跡することが可能となることから，*DF-Salvia* の適用範囲の拡大につながる．

今後は，本稿で述べた提案手法を実装し，既存のアプリケーションに適用することで，アクセス制御の精度とオーバヘッドの評価を行う．また，構造体や配列等の要素を考慮した field-positive なデータフロー解析手法について検討する．

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 Ver.1.4 http://www.jnsa.org/result/incident/data/2010incident_survey_PIL_v1.4.pdf

[2] 井田 章三，河島 裕亮，樫山 武浩，瀧本 栄二，毛利 公一：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，第 73 回全国大会講演論文集，Vol. 3，pp. 511-512，情報処理学会，2011.

[3] 河島 裕亮, 井田 章三, 樫山 武浩, 瀧本 栄二, 毛利 公一：DF-Salvia におけるアクセス制御のためのデータフロー解析手法，第 73 回全国大会講演論文集，Vol. 3，pp. 513-514，情報処理学会，2011.

[4] Alfred V.Aho, Ravi Sethi, Jeffrey D.Ullman 著，原田 賢一 訳：コンパイラ I/II 原理・技報・ツール，サイエンス社，1990.

[5] 樫山 武浩，河島 裕亮，井田 章三，瀧本 栄二，毛利 公一：データフロー情報に基づくアクセス制御のためのプログラム解析，情報処理学会研究報告，Vol. 2011-CSEC-52，No. 45，pp. 1-6，2011.