3Y-6

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

河島 裕亮[†]　井田 章三[†]　樫山 武浩[††]　瀧本 栄二[†††]　毛利 公一[†††]
[†] 立命館大学大学院 理工学研究科 [††] 立命館大学グローバル・イノベーション研究機構
[†††] 立命館大学 情報理工学部

1　はじめに

近年, プライバシ情報を電子データとして管理することが一般的である. そのため, 電子化されたプライバシ情報が, データ保有者の意図に反して漏洩する事件が多発している. 情報漏洩事件を引き起こす要因の多くは, プライバシ情報を扱うユーザの管理ミスによる流出やアプリケーションの誤操作, プライバシ情報を記録した媒体の紛失・置忘れが挙げられている. これらは, データへのアクセス権限を有する者によって情報漏洩が引き起こされているものであり, その件数は, 年々増加傾向にある. しかし, このような情報漏洩事件は, 暗号化や認証といった外部からの攻撃を防止することを目的としたセキュリティ技術で防止することが困難である.

以上の背景より, 我々は, 上述のような情報漏洩を防止することを目的としたオペレーティングシステム *DF-Salvia* の開発を行っている. 上述のような情報漏洩は, OS の視点から見るとシステムコールの発行を契機として発生する. そのため, *DF-Salvia* は, 情報漏洩を引き起こす要因となるシステムコールを監視・制御することでデータ保護を実現する. 特に, *DF-Salvia* は, プログラムの静的解析によるデータフロー情報生成 (以下, データフロー解析) の結果を用いて, データフロー単位でアクセス制御を行う点が特徴である. データフローを単位とすることでデータの発生源と使用箇所の関係を明確にし, 粒度の細かいアクセス制御を可能としている. 本稿では, *DF-Salvia* でアクセス制御を行うために必要なデータフロー解析手法と, それを OS へ通知する手法について述べる.

2　*DF-Salvia*

2.1　データフロー解析

DF-Salvia では, 情報漏洩を発生させる危険性のあるデータフローを単位としてアクセス制御を課す. そのデータフロー情報は, コンパイラなどのプログラムの静的解析によって生成することができる. 特に, *DF-Salvia* では, 定義使用連鎖を用いる.

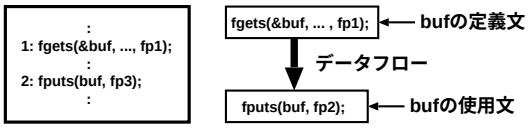

図 1　定義使用連鎖

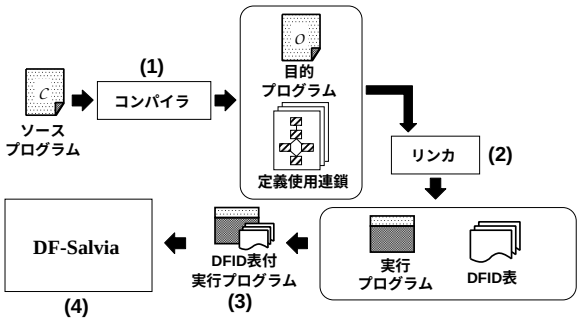

図 2　データフロー情報の通知過程

定義使用連鎖は, 変数に値を定義する命令文 (定義文) とその変数を使用する命令文 (使用文) の集合で表したものである (図 1). 定義文の代表的な例として, 代入文の左辺や, 引数に対してデータを格納する関数などが挙げられる. また, 使用点の代表例には, 代入文の右辺や条件式, 関数の引数などが挙げられる.

定義使用連鎖を解析することで, 保護ファイルから読み出されたデータがどの命令でどのように使用されるかを把握することが可能となる. すなわち, 保護すべき処理を把握することが可能となり, プロセスに対して過剰な制限を課すことなく, データを保護することが可能となる. 特に, *DF-Salvia* では, 図 1 に示すようなファイルからデータを読み出す命令文から読み出したデータを書き込むまでの流れに着目して定義使用連鎖を解析する.

なお今回は, コンパイラを改変することでデータフロー解析を行っている.

2.2　データフロー情報の生成と管理

データフロー解析と, その結果を OS に通知するまでの過程を図 2 に示す.

1. コンパイラを用いて定義使用連鎖解析し, 各連鎖に対してデータフロー ID を割り当てる.
2. データフロー ID 表 (以下 DFID 表) を作成する.
3. DFID 表と実行プログラムを一組で管理する.
4. OS は, プログラム実行時に DFID 表を取り出し, プログラムに対して制御を課すために用いる.

OS が, データフローを基にアクセス制御を課すためには, システムコールが属するデータフローを特定する

Data flow analysis for access control of *DF-Salvia*
Yusuke Kawashima[†], Shozo Ida[†], Takehiro Kashiyama[††], Eiji Takimoto[†††]and Koichi Mouri[†††]
[†]Graduate School of Science and Engineering, Ritsumeikan University
[††]Ritsumeikan Global Innovation Research Organization, Ritsumeikan University
[†††]College of Information Science and Engineering, Ritsumeikan University

必要がある．これは, 手順2で作成するDFID表が必要となる．DFID表は, 次の要素で構成する．

- データフローID
- ライブラリ関数の発行元アドレス
- 最終使用点を表すフラグ

DFID表を生成するためには, 1)read関数やfgets関数のような引数に値を格納するライブラリ関数を定義文として定義使用連鎖を解析する機能, 2) 各定義使用連鎖にIDを設定する機能, 3) 最終使用点を求める機能, 4)Call命令からCall命令までをフローとする定義使用連鎖を出力する機能, 5) 定義使用連鎖に含まれるライブラリ関数とその発行元アドレスを関連付ける機能の5つの機能が必要となる．また, 生成したDFID表をOSに通知するために, 6)DFID表を実行可能形式のファイルに格納する機能が必要となる．

*DF-Salvia*では, 情報漏洩を防止するためにファイルからデータを読み出す命令文から読み出したデータを書き込むまでの流れに着目して定義使用連鎖を生成する必要がある．しかし, 通常, プログラムのデータフロー解析を行うと, ライブラリ関数の引数は, 使用点として扱われる．そのため, 機能 1) によって, ファイルから読み出したデータを引数に格納するようなライブラリ関数を定義点として定義使用連鎖を解析する必要がある．

また, *DF-Salvia*で粒度の細かいアクセス制御を行うためには, 制御対象となるデータフローを容易に区別・管理するための情報と制御対象データフローを解放するための情報が必要となる．そのため, 機能 2) によってコンパイラで生成する各定義使用連鎖にIDを設定し, 機能 3) によって制御対象データフローの終了点として, 各定義使用連鎖の最後にあたる使用点 (最終使用点) を求め, データフロー情報としてOSに通知する必要がある．

さらに, *DF-Salvia*では, プログラム実行時に発行されたシステムコールが制御対象か否かを, そのシステムコールの発行元を特定し, その発行元が属するデータフローIDを求めることで判別する．そのため, 機能 4) によって, システムコールを発行する可能性のあるライブラリ関数から派生する定義使用連鎖を生成し, 機能 5) によってライブラリ関数の発行元アドレスを求め, 定義使用連鎖と関連付けを行う．Call命令の発行元アドレスは, リンク時に決定するため, リンカもしくは逆アセンブラを用いることで実現する．

1) 〜 5) の機能によって生成されたDFID表をOSに通知するためには, 実行プログラムと関連付けて管理する必要がある．そのため, 機能 6) によって, DFID表と実行プログラムを一組で管理するとともに, プログラム実行時に容易に参照できるようにする．

3 評価

今回, 2.2節で述べた機能を実現するために, 1) 〜 4) をCOINSコンパイラ [1] 内に実装し, 5) をobjdump内に実装した．また, 機能 6) は, objcopyの機能を用いることで実現した．

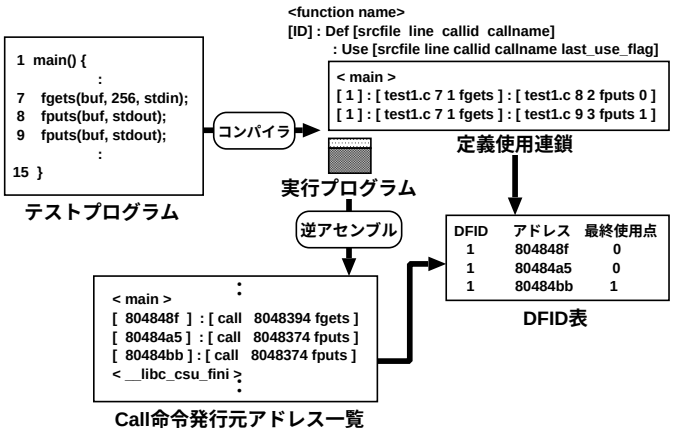

図3　テストプログラムを用いた各種生成物

上記の機能を実装したコンパイラで, 簡単なプログラムを対象にデータフロー解析および, DFID表を生成する実験を行った．実験の結果, 図3左上のプログラムをコンパイルすると, 同図右上の定義使用連鎖が生成された．この定義使用連鎖は, 各定義文・使用文が存在するソースプログラムや行番号, 関数名, 最終使用点を表すフラグといった情報を保持する．また, callidは, Call命令を識別するために割り当てる番号であり, 定義使用連鎖解析時に設定し, DFID表作成時に用いる．

次に, 改変したobjdumpを用いて実行プログラムを逆アセンブルすると, 図3左下のCall命令の発行元アドレス一覧が作成された．さらに, このCall命令発行元アドレス一覧とコンパイラで生成した定義使用連鎖を組み合わせて図3右下のDFID表が生成された．これにより, 定義使用連鎖の情報として含まれるcallidを基にして各定義点・使用点の発行元アドレスが特定されたことを確認できた．

また, 本機能をMicro Httpdと呼ばれる約300行程度の軽量なWebサーバを用いて実験を行ったところ, 適切なデータフロー解析を行い, 正しいDFID表が生成されることを確認した．これにより, ひとつの関数内でデータフローが閉じているプログラムに関してデータフロー情報を解析し, OSに通知する手法が実現できたことを確認した．

4 おわりに

本稿では, *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とデータフロー情報の通知方法について述べた．また, ひとつの関数内で閉じているデータフローに関してDFID表を作成する機能を実現したことを述べた．今後の課題として, ポインタや関数間のような大域的なデータフローに対応させる必要がある．

参考文献

[1] 中田育男, 渡邉坦, 佐々政孝, 森公一郎, 阿部正佳: “COINSコンパイラ・インフラストラクチャの開発,” コンピュータソフトウェア, Vol. 25, No. 1, pp. 2–18, 日本ソフトウェア科学会, 2008.