1L-6

コード変換を利用した機密度に基づくアクセス制御機構

奥野航平† 大月勇人† 瀧本栄二† 毛利公一†

† 立命館大学

1 はじめに

電子化された機密情報の漏洩事件が多発している．JNSA 2011 年情報セキュリティインシデントに関する調査報告書では，情報漏洩の原因として，アプリケーションの誤操作，データの管理ミス，外部記憶装置の紛失・置忘れ，プログラムのバグを挙げている．これらは，人為的ミス，すなわち，正当なアクセス権限を持つユーザにおいて発生しているといえる．認証や暗号化では，正当な権限を持つユーザは認証や復号を自由に行うことが可能であるため，それらを防止することが困難である．

以上の背景から，正当なアクセス権限を持つユーザによる情報漏洩を防止する User-mode DF-Salvia (以下，Salvia) を提案する．Salvia では，ユーザはファイルに対してコピー禁止などの機密度を設定できる．Salvia のアクセス制御機構は，機密度に基づいて計算機外部へのデータ出力処理を制御し，前述のような情報漏洩の防止を実現する．このアクセス制御を実現するには，データ出力時にデータの源となったファイルを特定する必要がある．この課題をデータの流れ (データフロー) の追跡により解決する．具体的には，ソースプログラムを書き換えるコード変換機構によりデータフローの動的な追跡を実現する．

2 User-mode DF-Salvia

2.1 データ保護を実現するアクセス制御

Salvia では，ユーザは保護したいファイルに対してデータの機密度を示した保護ポリシを付加できる．保護ポリシには，データの機密度として,「ファイルへの書出しの禁止 (コピー禁止) 」や「ネットワークへの送信の禁止」などを記述できる．

アクセス制御機構は，図 1 に示すように，プロセスによるファイルやネットワークへの出力をデータの機密度に基づき制御し，出力先がデータの機密度に違反するとき，その操作を禁止する．これにより，ユーザが意図しないデータの出力を防止し，情報漏洩を防止することができる．Salvia は，ユーザモードで動作し，OS に依存しないアクセス制御を実現する．そのため，既存のシステムを大きく変更せずにアクセス制御機構を導入できるというメリットがある．

2.2 構成

Salvia の構成を図 2 に示す．Salvia は，コード変換機構 (CTM) とアクセス制御機構 (ACM) の 2 つの機構から構成される．CTM は，プログラムのコードを変換するコンパイラであり，プログラムへアクセス制御に必要なコードを追加する．ACM は，アクセス制御を行うライブラリであり，コード変換したプログラムとリンクされ，実行時にアクセス制御を実現する．

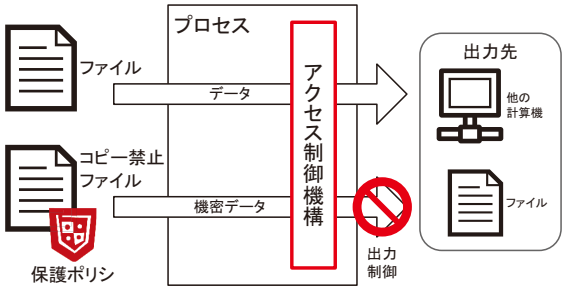

図 1: アクセス制御の概要

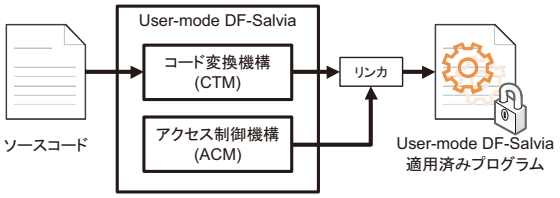

図 2: User-mode DF-Salvia の構成

3 コード変換

CTM は，コンパイル時のコード変換によりプログラムに対してデータフロー追跡機能とアクセス制御機能を追加する．データフロー追跡機能は，データの源となったファイルに関連付けられた保護ポリシを特定する機能である．アクセス制御機能は，データの出力処理を制御する機能である．

3.1 データフローの追跡

データフローは，ある領域のデータが別の領域にコピーされた時に発生するデータの流れである．Salvia は，このデータフローの追跡によってデータの源を特定し，動的テイント解析を用いてプログラムの実行時にデータフローを解析する．動的テイント解析は，データに対してタグと呼ぶ識別子を割り当て，データフローが発生したときにタグも同時に伝播させる．Salvia では，データフローからファイルを特定しやすくするためにタグとしてファイルを識別するための識別子 (FID) を割り当てる．FID は，UNIX におけるファイルディスクリプタや，C 言語の FILE 構造体へのポインタである．

ACM は，タグとデータが格納された領域のアドレ

An Access Control System Based On Confidential Level Using Code Transformation

Kohei OKUNO†, Yuto OTSUKI†, Eiji TAKIMOTO†and Koichi MOURI†

†Ritsumeikan University

スを関連付けて管理し，データフローが発生した時にコード変換されたプログラムから呼び出される．CTMは，データフローが発生するコードを変換し，次のデータフローを追跡することでプログラムのデータフローを網羅する．

- 代入処理
- 関数呼出し (引数・戻り値)
- コード変換対象外の関数呼出し
  (ライブラリ関数など)

データフローの他にも，データが直接コピーされずに情報が伝播する流れが存在し，これを暗黙的情報フローと呼ぶ．Salviaでは，暗黙的情報フローは対象としない．

3.1.1 代入処理

代入処理は，プログラム上の変数を別の変数にコピーする処理である．CTMは，代入処理の後にコードを追加し，実行時にACMが代入元の変数のアドレスから代入先の変数のアドレスに対してタグを伝播させる．

代入処理に演算を含む場合は，データが変形されているため，情報が変わったと判断し，タグの伝播を行わない．また，定数でデータが上書きされた場合は，タグの削除を行う．

3.1.2 関数呼出し

関数呼出しでは，関数の引数が暗黙的にコピーされる．また，呼出し元に戻るとき，戻り値としてデータが伝播する．これらを追跡するためにCTMは，関数を呼び出すコードの前後と関数の出入り口にコードを追加する．また，ACMは，タグを以下の手順で伝播させる．

1. 関数呼出し前に，実引数のアドレスとそのサイズをACMに通知する．
2. 関数の入口で，仮引数のアドレスとそのサイズをACMに通知し，ACMが手順1で得た情報を基にタグをコピーする．
3. 関数の出口で，戻り値として返される変数のアドレスとそのサイズをACMに通知する．
4. 関数呼出し元で戻り値が代入される前に，代入先のアドレスとそのサイズをACMに通知し，ACMが手順3で得た情報を基にタグをコピーする．

3.1.3 コード変換対象外の関数呼出し

ライブラリ関数などコード変換の対象外となる関数は，データフローの追跡ができない．これらは，関数呼出しをコード変換ルールにしたがって変換し，コードを追加する．コード変換ルールには，関数呼出し後に与えられた引数や戻り値でどのようにデータが伝播するかを指定する．

また，ファイルからデータを読み出す関数はデータの起点となるため，読み出されたデータに対してタグを付加する．この関数呼出しも同様にコード変換ルールにしたがって変換する．

コード変換ルールは，ライブラリ関数の仕様に基づいて作成する．一度作成したコード変換ルールは，同じライブラリ関数を使用するプログラムに再利用できる．

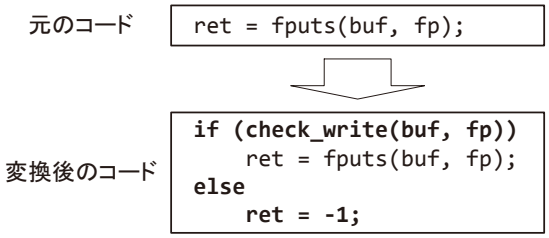

図 3: アクセス制御の擬似コード

3.2 ファイル I/O の制御

保護ポリシに従ったアクセス制御を行うためには，データの出力先の識別が必要になる．ユーザモードでは，FIDから直接出力先を判別できない．そこで，fopen関数などのFIDを生成したときに使用した関数の引数や戻り値の情報から出力先を識別する．これらの情報は，ACMで動的に関連付けて管理される．また，CTMは，FIDを生成する関数呼出しをコード変換ルールにしたがってコードを追加する．

CTMにより追加されるコードの例を図3に示す．この例は，ファイルへデータを書き出すfputs関数を対象に制御用のコードを追加した擬似コードである．ファイルI/O処理は，それらの関数呼出しを対象にコード変換を行い，追加した条件分岐を用いて処理を分岐させ，ファイルI/O処理のスキップにより制御を実現する．条件式には，ACMのアクセスを判断するcheck_write関数の戻り値を使用する．また，ACMがアクセスを禁止した場合，本来実行される関数の実行が失敗したときのエラー値を戻り値に代入する処理を行う．これにより，プログラムにアクセス制御の動作を通知し，プログラムの例外処理による継続した動作を可能にする．

4 実装と検証

CTMは，LLVMのパスを拡張し，中間表現を変換する機構として実装した．ACMは，共有ライブラリとして実装した．

以上のアクセス制御機構の動作検証を行うために，オープンソースソフトウェアをコンパイルした．検証に使用したソフトウェアは，cp，ftp，mailコマンドとperlインタプリタであり，それぞれ保護ポリシに従ったアクセス制御が機能することを確認した．

Salviaを適用したcpコマンドを用いてファイルコピーのスループットを計測した結果，約25 %の低下となった．しかし，スループット自体は，約1.6 GB/sと高速であるため実用上問題とならない．また，perlインタプリタ上でCGIスクリプトを動作させて遅延を比較した結果，約7倍となった．これは，CPUバウンドな処理においての動的テイント解析のオーバヘッドが表れている．

5 おわりに

本論文では，人為的なミスによる情報漏洩を防止するSalviaについて述べた．Salviaは，コード変換によるアクセス制御機能の追加によって，データの機密度に基づくアクセス制御を実現し，情報漏洩を防止する．