Vol.2016-DPS-166 No.33
Vol.2016-CSEC-72 No.33
2016/3/4

Argosの動的テイント解析機能を利用したOSによる情報漏洩防止手法

松本 隆志[1,a)]　大月 勇人[1]　明田 修平[1]　瀧本 栄二[1]　齋藤 彰一[2]　毛利 公一[1]

概要：情報漏洩インシデントの主な原因は，人為的なミスによるものであると報告されている．本論文では，そのような人為的なミスによる情報漏洩を防止するセキュアシステム TA-Salvia を提案する．TA-Salvia は，「Argos が有する動的テイント解析機能を OS が活用」して，ユーザプロセスが扱うデータのフローを追跡し，漏洩を検出・防止するところが特徴的である．データの使用許可範囲を設定し，その範囲を超えた場合に漏洩と判定するために，TA-Salvia ではファイル毎に保護ポリシを設定可能としている．本論文では，TA-Salvia のコンセプトについて提案し，その設計と実装について述べる．また，インターネット上で公開されている実アプリケーションを使用して評価を行ったので，その結果について述べる．

キーワード：情報漏洩防止，ファイルアクセス制御，動的テイント解析，オペレーティングシステム

A Data Loss Prevention Method by OS Using Dynamic Taint Analysis Function of Argos

Takashi Matsumoto[1,a)]　Yuto Otsuki[1]　Shuhei Aketa[1]　Eiji Takimoto[1]　Shoichi Saito[2]
Koichi Mouri[1]

Abstract: Many data loss incidents by human error have been reported. This paper proposes a data loss prevention system, called TA-Salvia, caused by human error. TA-Salvia uses dynamic taint analysis function of Argos to track data flow in user processes. Users can specify data distribution scope in data protection policy. TA-Salvia prevents data loss based on the policy. This paper proposes about concept of TA-Salvia and describes its design and implementation. TA-Salvia is evaluated by real applications published on the Internet and we confirmed data loss prevention.

Keywords: Data Loss Prevention, File Access Control, Dynamic Taint Analysis, Operating System

1. はじめに

情報システムの発展によって個人情報が電子化されるようになり，電子化された個人情報の漏洩事件が増加している．JNSA 2013 年情報セキュリティインシデントに関する調査報告書 [1] によると情報漏洩の発生件数の約 80 %は，「管理ミス」，「誤操作」および「紛失・置き忘れ」が原因となっている．「管理ミス」，「紛失・置き忘れ」は，管理者が個人情報を記憶媒体に保存し，外部に持ち出したことが原因となっている事例が多く，「誤操作」は，管理者が個人情報を間違えてメールに添付し，送信してしまうといった事例が多い．これは，コンピュータウイルスやハッキングによる不正アクセスではなく，正当なアクセス権限を持つユーザの人為的なミスによる情報漏洩であるという特徴がある．

情報漏洩を防止するための既存のセキュリティ技術として，ユーザ認証によるアクセス制御やファイルの暗号化などが存在する．しかし，これらの技術を利用した場合にお

[1] 立命館大学
Ritsumeikan University, Kusatsu, Shiga 525–8577, Japan
[2] 名古屋工業大学
Nagoya Insitute of Technology, Nagoya, Aichi 466–8555, Japan
[a)] tmatsumoto@asl.cs.ritsumei.ac.jp

いて，正当なアクセス権限を持ったユーザが認証・復号した後は，自由にデータが扱えるため，人為的ミスによる情報漏洩の防止が困難である．また，既存のセキュリティ技術を拡張したシステムとして，SELinux[2] や TOMOYO Linux[3] などの強制アクセス制御がある．強制アクセス制御は，プロセスのリソースに対する操作毎にアクセス権限を詳細に設定できるため，より強力なアクセス制御を行うことができる．事前にファイルの機密度が固定的に決まっている場合には，それに適したセキュリティポリシを記述することで，人為的ミスによる情報漏洩の防止が可能である．しかし，ファイルの機密度が変化する場合などについては，それに適応したセキュリティポリシを記述することはできない．また，従来のアクセス制御で使用されていた ACL やパーミッションなどのファイルに対して設定するポリシの他に，多種類のポリシを必要とするため，データの保護を目的とする設定の記述が複雑化しやすいといった問題がある．特に，一般ユーザが個々に決定するのは困難である．

このような問題を解決する先行研究として，著者らは，Salvia Linux[4]，DF-Salvia[5]，User-mode DF-Salvia[6] を提案した．これらは，データ提供者の意図する方針（機密度に相当する）をデータ保護ポリシ（以下，ポリシ）として定義し，ファイル毎に設定可能とする．ポリシが設定されたファイルは，ポリシの記述に従って制御される．Salvia Linux は，すべてのアプリケーションが透過的に制御される．ただし，Salvia Linux は，プロセスを主体としてアクセス制御を行うため，過剰なアクセス制御が発生してしまう．DF-Salvia と User-mode DF-Salvia は，Salvia Linux で発生する過剰なアクセス制御の問題を解決している．ただし，事前にコンパイラを用いてプログラムのデータフローを解析またはコードを変換する必要があるため，システム全体のアプリケーションに共通に適用されるわけではない．

そこで，本論文では，Argos[7] を用いてデータを追跡することで，過剰なアクセス制御の問題を解決し，かつシステム全体のアプリケーションに適用可能な情報漏洩防止システム TA-Salvia を提案する．Argos は，動的テイント解析機能を有する x86 エミュレータである．TA-Salvia は，Argos の動的テイント解析機能を OS が利用することでアクセス制御を実現する．Argos は，メモリに書き込まれた保護データを追跡し，OS は，ファイル書込み時に追跡結果を Argos から取得する．OS は，追跡結果から保護データの源となるファイルに設定されたポリシを特定し，ポリシに基づいたアクセス制御を行う．Argos は，バイト単位でデータを識別し，追跡することが可能である．そのため，TA-Salvia は，バイト単位の粒度の細かいアクセス制御が可能である．また，メモリ上のデータを追跡・制御するため，アプリケーションに変更を加えることなく適用可能である．

以下，2 章で先行研究について述べ，3 章で提案手法を述べる．4 章で Argos について述べ，5 章と 6 章で TA-Salvia の実装について述べる．7 章で評価，8 章で関連研究について述べ，9 章でまとめる．

2. 先行研究

先行研究として著者らは，正当なアクセス権限を持つユーザによる人為的なミスを防止するセキュアシステム Salvia Linux を提案した．Salvia Linux は，データ提供者が意図する保護方針をポリシとして定義し，保護対象のファイルと組にして管理する．ポリシは，計算機やプロセスの状況を示すパラメータ（ユーザ ID，時刻，IP アドレスなど）を用いて記述できる．例えば，「社内の IP アドレスにのみ送信を許可する」と記述されたポリシを設定することで，ネットワークによる情報漏洩を防止することができる．また，「データ提供者以外のユーザは，読出し可能であるが，書込みや他のプロセスとのデータ共有を禁止する」と記述されたポリシを設定することで，提供者以外のユーザは，ファイルを閲覧できるが，ファイルの改竄や記憶媒体への書込みができないといった制御が可能である．Salvia Linux は，保護ファイルのデータを読み出したプロセスに対し，ポリシに基づいた強制アクセス制御を課すことで，プライバシを考慮したデータ保護を実現している．具体的には，プロセスの挙動を監視し，データ漏洩が発生する可能性のある計算機資源へアクセスしたとき，当該プロセスが過去に読み出した全保護ファイルのポリシに違反しないアクセスのみを許可する．すなわち，プロセスを主体としたアクセス制御を行う．Salvia Linux は，このようなデータ保護機構を OS 内に有しているため，すべてのアプリケーションに透過的に適用可能である．ただし，Salvia Linux は，プロセスを主体としたアクセス制御を行っているため，過剰なアクセス制御が発生するという問題がある．

先行研究の DF-Salvia と User-mode DF-Salvia は，Salvia Linux の過剰なアクセス制御の問題を解決している．DF-Salvia は，コンパイラと協調し，プロセス内の個々のデータフローを主体として管理・制御することでアクセス制御を行う．User-mode DF-Salvia は，プログラムへアクセス制御用のコードを追加することで，プログラム単体でアクセス制御が可能となり，プラットフォームに依存しないアクセス制御を実現する．User-mode DF-Salvia は，プログラムのデータフローに基づいてデータを区別している．DF-Salva と User-mode DF-Salvia は，いずれも事前にコンパイラを用いてプログラムのデータフローを解析またはコード変換をする必要があり，システム全体のアプリケーションに共通に適用させるにはコストが高い．

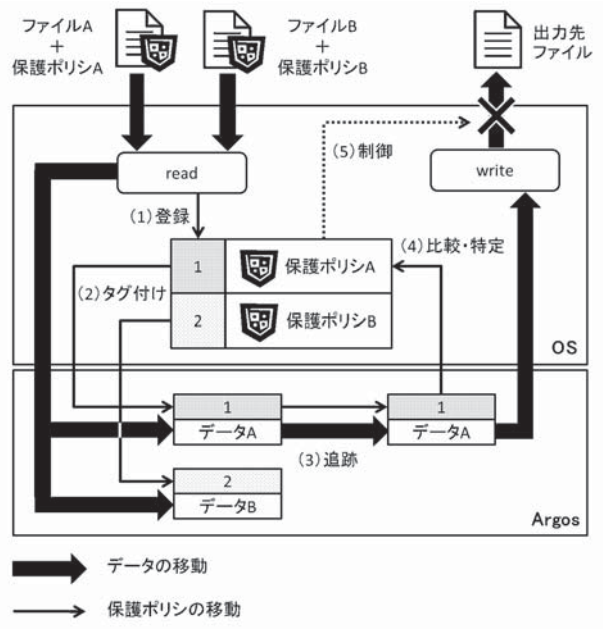

図 1　TA-Salvia のアクセス制御モデル

3. 提案手法

本論文では，先行研究の問題を解決する情報漏洩防止システム TA-Salvia を提案する．本章では，まず動的テイント解析について述べ，続いて提案手法のアクセス制御モデルとその動作手順について述べる．

3.1 動的テイント解析

動的テイント解析は，解析したいデータに対してテイントタグ（以下，タグ）と呼ばれる識別子を割り当て，タグと共にデータの伝播を追跡する技術である．動的テイント解析は，タグを割り当てたデータが別の領域に伝播されるとき，タグも同時に伝播させる．データに付いたタグを確認することで，データフローの識別が可能である．

Argos は，動的テイント解析機能を CPU エミュレータに実装している．TA-Salvia は，Argos が持つ動的テイント解析機能を利用することでアクセス制御を行う．Argos については 4 章で述べる．

3.2 アクセス制御モデル

TA-Salvia のアクセス制御モデルを図 **1** に示す．TA-Salvia は，Argos の動的テイント解析機能を OS が利用することでアクセス制御を実現する．Argos は，動的テイント解析を用いて物理メモリ上のデータを追跡する．OS は，ファイル読出し時にファイルのデータを Argos に追跡させる．そして，ファイル書込み時に Argos から追跡した結果を受け取りアクセス制御に利用する．TA-Salvia は，以下の手順でアクセス制御を行う．

(1) 登録
OS は，ファイル読出し時にファイルに設定されたポリシを取得する．取得したポリシは，タグ番号が割り当てられ，OS に登録される．複数のファイルを読み出した場合，それらに設定されたポリシは，それぞれ別のタグ番号が割り当てられる．

(2) タグ付け
登録が完了すると，OS は，タグ番号と読み込んだファイルが格納されているメモリ領域の情報を Argos に渡す．これにより，OS は，読み出したデータを Argos に追跡させることができる．

(3) 追跡
Argos は，OS から受け取った情報を基にデータの追跡を開始する．

(4) 比較・特定
OS は，ファイル書込み時にそのデータが格納されている領域と対応するタグ番号を Argos から取得する．OS は，Argos から取得したタグ番号からデータの源となるファイルに設定されたポリシを特定する．

(5) 制御
書込むデータは，特定したポリシに従って制御される．

4. Argos

Argos は，ゼロデイ攻撃の検知を目的とし，ハードウェアエミュレータである QEMU[8] を改良したものである．Argos は，外部から入力されるデータを汚染データとみなし，動的テイント解析を用いて汚染データを追跡する．

4.1 Argos における動的テイント解析

Argos は，動的テイント解析を実現するために物理メモリとレジスタに対して 1 対 1 に対応する動的テイント解析用の物理メモリ（シャドウメモリ）とレジスタ（シャドウレジスタ）の記憶領域を用意している．Argos 本来の動作では，ネットワークカードで受け取ったデータが物理メモリに書き込まれるとき，書込み先のアドレスに対応する動的テイント解析用のメモリ領域にタグを付加する．CPU が命令をフェッチするたびにタグを確認し，テイント伝播ルールに基づいて伝播させる．

4.2 タグの管理

Argos のタグは，1 バイトのデータに対して汚染されているかどうかを 1 バイトフラグで表される．タグは，レジスタやメモリに 1 対 1 で対応したシャドウ領域（シャドウレジスタとシャドウメモリ）に保存する．レジスタとメモリのタグ管理方法については，以下で述べる．

レジスタ　x86 アーキテクチャには，8 つの汎用レジスタがある．Argos は，各レジスタにタグを付けるための領域を 4 バイトずつ確保する．これをシャドウレジス

タと呼ぶ．x86 アーキテクチャには，他にもセグメントレジスタや EFLAGS のような特殊なレジスタが存在する．しかし，Argos でタグを管理する必要があるのは，データが移動するレジスタのみである．そのため，汎用レジスタ以外には，タグ付けを行わない．

物理メモリ　ストア命令により，汚染データが物理メモリに書き込まれた場合，シャドウレジスタからタグを移動させなければならない．そのため，Argos は，タグ管理のために物理メモリと 1 対 1 に対応したバイトマップを用意している．これをシャドウメモリと呼ぶ．レジスタから物理メモリへデータが移動した場合，タグも同時にシャドウレジスタからシャドウメモリへ移動させる．

5. Argos の拡張

TA-Salvia では，OS がタグを登録・取得するためのインタフェースを追加する必要がある．本章では，Argos に追加したインタフェースとその動作について述べる．なお，TA-Salvia では，Argos-0.7.0 をベースに拡張した．

5.1 仮想 PCI デバイスの追加

TA-Salvia は，Argos の動的テイント解析機能を OS が利用することでアクセス制御を行う．しかし，Argos は，OS にタグを登録・取得させるための機能を持たない．そのため，TA-Salvia では，OS がタグを登録・取得するためのインタフェースとして，Argos に仮想 PCI デバイス（salvia_device）を追加した．また，OS には，そのデバイスを管理するデバイスドライバ（salvia_driver）を追加した．OS は，salvia_driver を通して Argos の salvia_device に要求を伝え，Argos は，salvia_device を通して OS の salvia_driver に追跡結果を伝える．

salvia_driver と salvia_device 間の通信は，MMIO 領域を通して行う．salvia_driver は，MMIO 領域にデータの追跡要求を書き込み，MMIO 領域に書き込まれたデータは，4 バイトずつ salvia_device に送信される．

5.2 タグ情報

salvia_device へのタグ付け・取出し処理では，タグ情報を用いる．タグ情報の構成を以下に示す．

物理アドレス　タグ付け先またはタグ取出し先の物理アドレスを指定する．

データサイズ　タグ付けするデータのサイズを指定する．タグ取出し時には，使用しない．

タグ番号　シャドウメモリに実際に書込む番号を指定する．Argos は，ここで指定した番号を用いてデータの追跡を行う．タグ取出し時には，使用しない．

コマンド　salvia_device の動作を指定する．コマンドが 1 の時にタグ付け処理を行い，2 の時にタグ取出し処理を行う．

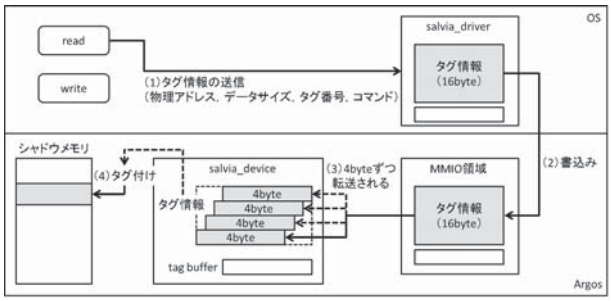

図 2　タグ付け手順

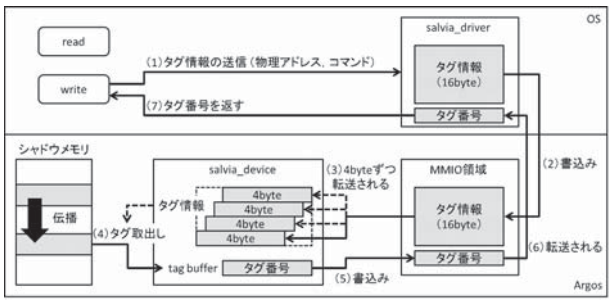

図 3　タグ取出し手順

salvia_device は，タグ情報を基にシャドウメモリへのタグ付け・取出し処理を行う．

5.3 タグ付け手順

salvia_device は，ファイル読出し時に salvia_driver からタグ付け要求を受け取る．タグ付け要求を受け取った salvia_device は，タグ情報を基にシャドウメモリにタグを付ける．タグ付け手順を**図 2** に示す．salvia_device は，以下の手順でタグ付けを行う．

(1) ファイル読出しが発生したとき，タグ付けに必要な情報を salvia_driver に送信する．タグ付けに必要な情報は，物理アドレス，データサイズ，タグ番号，コマンドの 4 つである．

(2) salvia_driver は，受け取った情報からタグ情報を生成し，MMIO 領域に書き込む．

(3) MMIO 領域に書き込まれたタグ情報は，16 バイトなので 4 バイトずつ salvia_device に送信される．

(4) タグ情報を受け取った salvia_device は，タグ情報を基にシャドウメモリにタグ付けを行う．

5.4 タグ取出し手順

salvia_device は，ファイル書込み時に salvia_driver からタグ取出し要求を受け取る．タグ取出し要求を受け取った salvia_device は，タグ情報を基にシャドウメモリからタグ番号を取得する．取得したタグ番号は，salvia_driver に返される．タグ取出し手順を**図 3** に示す．salvia_device は，以下の手順でタグ取出しを行う．

(1) ファイル書込みが発生したとき，タグ取出しに必要な

情報を salvia_driver に送信する．タグ取出しに必要な情報は，物理アドレス，コマンドの 2 つである．

(2) salvia_driver は，受け取った情報からタグ情報を生成し，MMIO 領域に書き込む．

(3) MMIO 領域に書き込まれたタグ情報は，16 バイトなので 4 バイトずつ salvia_device に送信される．

(4) タグ情報を受け取った salvia_device は，タグ情報を基にシャドウメモリからタグ番号を取り出す．取り出したタグ番号は，tag buffer に格納される．

(5) salvia_device は，tag buffer のタグ番号を MMIO 領域に書き込む．

(6) MMIO 領域に書き込まれたタグ番号は，salvia_driver に送信される．

(7) salvia_driver は，受信したタグ番号をファイル書込みを行う関数に返す．

6. TA-Salvia のアクセス制御

TA-Salvia は，ファイル読出しやファイル書込みを制御することでアクセス制御を実現する．TA-Salvia は，read /write システムコールと mmap システムコールによるファイルアクセスに対応している．TA-Salvia は，これらの処理をフックし，アクセス制御用の処理を追加している．本章では，ポリシの管理方法について述べ，ファイルデータの制御方法について述べる．

6.1 保護ポリシの管理

読み出したファイルに設定されたポリシを管理するためのテーブルとして policy_table を OS に用意している．policy_table は，次の要素から成る．

- タグ番号
- 保護ポリシ

ファイル読出し時に OS は，タグ番号を割り当てて policy_table にポリシを登録する．また，ファイル書込み時に Argos からデータの追跡結果としてタグ番号を取得した際に，タグ番号からファイルに適用すべきポリシを取得する．

6.2 ファイル読出し禁止

ファイル読出し禁止の判定は，read システムコールまたは mmap システムコール実行時に行う．TA-Salvia は，まずファイルに設定されたポリシを取得する．ポリシが設定されていない場合は，元のシステムコールの処理に戻る．ファイル読出し禁止のポリシが設定されていた場合，ファイル読出しを実行せずにエラーを返す．

6.3 ファイル書込み禁止

6.3.1 データ追跡開始

ファイル書込み禁止のポリシが設定されたファイルは，データの追跡をする必要がある．read システムコールで読み出されたファイルは，read システムコール実行後にデータの追跡を開始する．また，mmap システムコールで読み出されたファイルは，ページフォルト実行時にデータの追跡を開始する．具体的には，ページフォルト処理により，ページテーブルが更新され，プロセスが仮想アドレスを用いてファイルにアクセス可能となった後，データ追跡に必要な処理を実行する．

TA-Salvia は，まずファイルに設定されたポリシを取得する．ポリシが設定されていない場合は，read システムコールまたはページフォルトの処理に戻る．ファイル書込み禁止のポリシが設定されていた場合は，データの追跡が必要である．データの追跡のため，TA-Salvia は，ファイルに設定されたポリシを policy_table に登録する．登録後，TA-Salvia は，タグ情報を送信し，Argos のシャドウメモリにタグを付ける．タグ付けされたデータは，Argos により追跡される．

6.3.2 ファイル書込み禁止の判定

ファイル書込み禁止の判定は，write システムコールまたはディスク書込み実行時に行う．ディスク書込みは，カーネルが定期的にバックグラウンドで実行またはファイル同期関連のシステムコール発行時に行われる．

TA-Salvia は，まず書き込むデータがポリシを適用すべきデータかどうかを確認する．ポリシの特定には，タグ番号が必要である．そのため，TA-Salvia は，タグ情報を送信し，Argos から書き込むデータと対応するタグ番号を取得する．タグ番号が 0（タグなし）の場合，ファイル書込みまたはディスク書込みを実行する．タグ番号が 1 以上（タグあり）の場合は，policy_table とタグ番号を比較し，ポリシを特定する．書込み禁止以外のポリシが設定されていた場合，通常通りファイル書込みが実行される．書込み禁止のポリシが設定されていた場合，ファイル書込みまたはディスク書込みを実行せずにエラーを返す．

7. 評価

本章では，ユーザプロセスが扱うデータフローを追跡し，ファイルのポリシに従ってアクセス制御ができることを検証する．インターネット上で公開されている実アプリケーションを用いて動作検証を行った．検証に用いるアプリケーションは，以下の 2 つである．

- Emacs
  Emacs は，高機能なテキストエディタである．ユーザは，エディタを用いて機密ファイルからデータの一部をコピーし，別のファイルにペーストして使用する場合がある．このとき，TA-Salvia でファイルの保存を制限できるかを検証する．
- Mailx
  Mailx は，電子メールの送受信を行うアプリケーションである．TA-Salvia でファイルの誤送信を防止でき

Vol.2016-DPS-166 No.33
Vol.2016-CSEC-72 No.33
2016/3/4

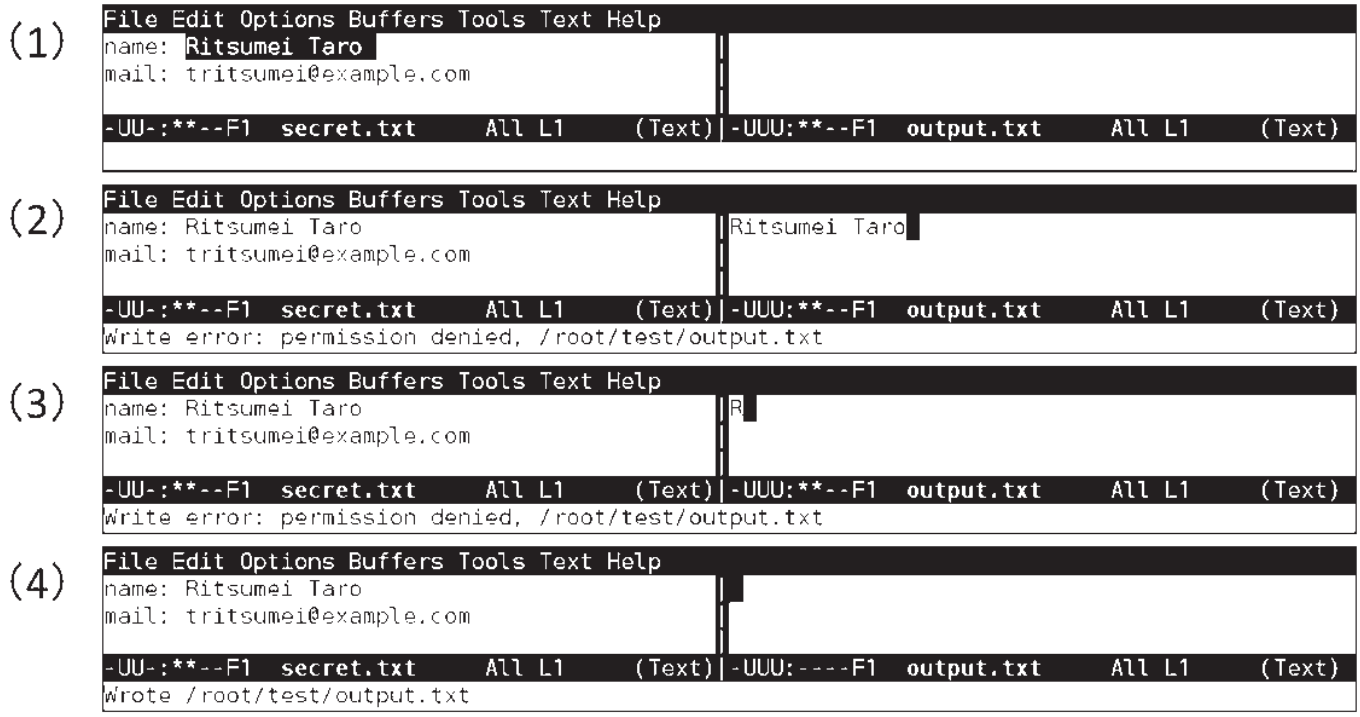

図 4　Emacs を用いた動作検証の結果

るかを検証する．Mailx は，バイナリファイル添付時に BASE64 を用いてエンコードするため，暗黙的フローが発生する．

7.1　Emacs を用いた動作検証

Emacs を用いた動作検証のために，他のファイルへの書込みを禁止するポリシが設定された機密ファイル（secret.txt）とファイルへのアクセスをすべて許可するポリシが設定された出力ファイル（output.txt）を用意した．Emacs を用いて，secret.txt のデータの一部をコピーし，output.txt にペーストする．この動作検証では，機密ファイルのデータがバイト単位で識別され，そのデータの書込みを制限できるかを確認する．TA-Salvia によるアクセス制御が成功した場合，出力ファイルに機密ファイルのデータが 1 バイト以上存在している間は，ファイルの保存に失敗する．

Emacs を用いた動作検証の結果を**図 4** に示す．この動作検証では，Emacs の機能を用いて画面の分割をしている．左は，secret.txt の内容が表示されており，右は，output.txt の内容が表示されている．

(1) secret.txt からデータの一部「Ritsumei Taro」を選択し，コピーした．
(2) コピーしたデータを output.txt にペーストし，保存した．「Write error: permission denied, /root/test/output.txt」とエラーが表示されていることから，保存に失敗していることが分かる．
(3) ペーストしたデータの一部「itsumei Taro」を削除し，「R」のみを残した状態で，保存した．しかし，同様のエラーが表示されていることから，保存に失敗していることが分かる．
(4) ペーストしたデータをすべて削除した状態で保存した．「Wrote /root/test/output.txt」と表示されていることから，保存に成功したことが分かる．

以上から，機密ファイルのデータは，バイト単位で識別され，TA-Salvia により，保存が制限されていることが確認できた．

7.2　Mailx を用いた動作検証

Mailx を用いた動作検証のために，他のファイルへの書込みを禁止するポリシが設定されたテキストファイル（secret.txt）と同様のポリシが設定された画像ファイル（secret.png）を用意した．Mailx を用いて，これらのファイルをメールに添付して送信する．Mailx は，バイナリファイルを添付した場合，BASE64 エンコードのために暗黙的フローを発生させる．この動作検証では，暗黙的フローが発生した場合，TA-Salvia のアクセス制御を正しく行えるかを確認する．TA-Salvia によるアクセス制御が成功した場合，テキストファイルや画像ファイルが添付されたメールの送信は，すべて失敗するはずである．

Mailx を用いた動作検証の結果を**図 5** に示す．3 行目では，Mailx を用いて，テキストファイルが添付されたメールを送信している．その結果，7-8 行目で「Permission denied」とエラーが表示された．また，10 行目で「. . . message not sent.」と表示されており，メールの送信が失敗していることが分かる．

```
1  root@salvia:~/test# ls
2  secret.png  secret.txt
3  root@salvia:~/test# mail -a secret.txt -s '
       TextFileTest' tmatsumoto@asl.cs.
       ritsumei.ac.jp
4  This is test mail.
5  .
6  EOT
7  temporary mail file: Permission denied
8  Permission denied
9  "/root/dead.letter" 1/19
10 . . . message not sent.
11 root@salvia:~/test# mail -a secret.png -s '
       ImageFileTest' tmatsumoto@asl.cs.
       ritsumei.ac.jp
12 This is test mail.
13 .
14 EOT
```

図 5 Mailx を用いた動作検証の結果

11 行目では，Mailx を用いて，画像ファイルが添付されたメールを送信している．エラーが表示されていないことから，メールの送信が成功していることが分かる．以上から，テキストファイルが添付されたメールの送信は，TA-Salvia で防止できた．しかし，画像ファイルが添付されたメールの送信は，TA-Salvia で防止できなかった．Mailx は，バイナリファイル添付時に BASE64 を用いてエンコードするため，暗黙的フローが発生する．そのため，画像ファイルを添付したメールの送信は，暗黙的フローによりタグの伝播が途切れてしまい，TA-Salvia で制御できなかったと考えられる．

8. 関連研究

8.1 動的テイント解析

Argos は，TA-Salvia のような情報漏洩防止システムに拡張されてきた以外に，マルウェアのコードを解析するためにも拡張されてきた．API Chaser[9] は，解析対象のマルウェアのプログラムコードにタグを付与し，追跡することでコードの識別を可能にする．API Chaser は，Argos を用いて追跡するため，マルウェアが他のプロセスにプログラムを注入したとしてもコードの識別が可能である．API Chaser は，ディスク上のファイルに対してもタグを保持できるよう，ファイルに関するタグ保持用のデータ構造体，シャドウディスクを Argos に追加実装している．API Chaser は，OS がブートされる前に，シャドウディスクに対してタグを付けることで対象のコードを追跡することができる．

ハードウェアエミュレータを用いた動的テイント解析は，Argos 以外も存在する．バイナリ解析プラットフォーム BitBlaze[10] の構成要素の 1 つに TEMU があり，これも，Argos と同様に QEMU を拡張し，動的テイント解析機能を追加している．TEMU を用いた研究として Panorama[11] がある．Panorama は，TEMU をベースとしたシステムで，ユーザのプライバシを侵害するマルウェアの検知が目的である．Panorama は，マルウェアに検知されないようにハードウェア（キーボードやハードディスクなど）からの入力をフックし，タグ付けを行う．また，Panorama は，スワップアウトした場合でも追跡できるようにハードディスクに対してタグ管理用のシャドウ領域を用意している．

API Chaser や Panorama のようにプログラムの挙動を解析するために動的テイント解析を用いた例が多い．TA-Salvia は，これを情報漏洩の防止のために OS と連携し，ファイルアクセス制御に適用している点で新たな試みである．

8.2 情報漏洩防止

TaintEraser[12] は，Intel が開発した Pin[13] を利用して動的テイント解析を行い，情報漏洩を防止する．Pin は，実行ファイルに対してコード挿入することで，実行時の情報の取得する．Pin は，このような情報取得ための様々な API を提供している．TaintEraser は，レジスタやメモリにアクセスする命令を Pin が提供する命令解析 API を用いてフックし，動的テイント解析に利用する．TaintEraser は，ファイルやキーストロークに対してプライバシポリシを設定し，それらがファイルやソケットに出力される時，ランダムなデータに置き換えることで情報漏洩を防止する．TaintEraser は，事前に Pin を用いて実行ファイルを解析する必要があるため，すべてのアプリケーションに適用することが困難である．一方，TA-Salvia は，Argos を用いて物理メモリ上のデータを追跡し，アクセス制御を行うため，すべてのアプリケーションに適用可能である．

栗田らは，プログラム実行時の情報フローを動的に追跡，制御するプラットフォームを開発し，それを用いた情報漏洩の防止手法を提案している [14]．このプラットフォームは，x86 エミュレータである Bochs[15] をベースに開発されており，動的テイント解析機能を持つ．このプラットフォームは，出力制限データを起点とする暗黙的フローが発生する期間を区別し，その期間を保護観察モードとして管理する．保護観察モード中に出力されるデータは，保護ポリシに従って制御される．ただし，保護観察モードの終了宣言は，プログラマが行うため，アプリケーションに変更を加える必要がある．また，栗田らは，情報フローの追跡・制御に必要となるプロセッサ機能の実装のみ行っており，アクセス制御に必要な機能を OS に実装していない．TA-Salvia は，暗黙的フローに対応できないが，すべてのアプリケーションに適用可能である．また，TA-Salvia は，read/write システムコールと mmap システムコールによるファイルアクセスを制御し，実アプリケーションを用い

て情報漏洩が防止できることを確認している．

EDP（Enterprise Data Protection）[16] は，Microsoft が開発している企業のデータ保護を目的としたシステムである．EDP は，企業が使用するアプリケーション（以下，企業アプリ）と個人が使用するアプリケーション（以下，個人アプリ）を分割して管理する．EDP は，企業アプリに対してポリシを設定し，企業アプリのデータが個人アプリで使用できないよう制限する．また，企業アプリで作成したファイルは，すべて暗号化して管理される．そのため，他の端末に送信されたとしても利用できないといった特徴を持つ．EDP は，企業アプリに対してポリシを設定している．そのため，企業アプリで作成したファイルは，すべて制限されてしまう．TA-Salvia は，ファイル毎にポリシを設定するため，ファイル単位で制御することができる．

9. おわりに

本論文では，Argos の動的テイント解析機能を OS で利用することで情報漏洩を防止する TA-Salvia について述べた．TA-Salvia は，Argos を用いて物理メモリ上のデータを追跡し，その追跡結果を OS 側で利用することでアクセス制御を実現する．TA-Salvia は，データをバイト単位で識別してアクセス制御を行うため，Salvia Linux で問題となっていた過剰なアクセス制御が発生しない．また，TA-Salvia は，物理メモリ上のデータを追跡するため，システム全体のアプリケーションに適用可能である．

評価では，実アプリケーションに対して TA-Salvia がポリシに基づいて機密情報の出力を禁止し，情報漏洩を防止できることを示した．ただし，バイナリファイルを添付したメールを Mailx で送信した場合は，暗黙的フローが発生するため，TA-Salvia によるアクセス制御が正しく行えなかった．今後は，暗黙的フローについて検討する必要がある．また，現在の実装では，read/write システムコール，mmap システムコールによるファイルアクセスに制限を課している．しかし，ネットワーク関連のシステムコールについては，まだ制御できていないため，今後実装する必要がある．

参考文献

[1] NPO 日本ネットワークセキュリティ協会：2013 年情報セキュリティインシデントに関する調査報告書〜個人情報漏えい編〜第 1.2 版，`http://www.jnsa.org/result/incident/` (2015).
[2] Loscocco, P. and Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System, *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pp. 29–42 (2001).
[3] 原田季栄，保理江高志，田中一男：TOMOYO Linux - タスク構造体の拡張によるセキュリティ強化 Linux，*Proceedings of the Linux Conference 2004*, pp. 1–10 (2004).
[4] 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(ACS13), pp. 1–15 (2006).
[5] 内匠真也，奥野航平，大月勇人，瀧本栄二，毛利公一：コンパイラと OS の連携によるデータフロー追跡手法，情報処理学会論文誌，Vol. 56, No. 12, pp. 2313–2323 (2015).
[6] 奥野航平，内匠真也，大月勇人，瀧本栄二，毛利公一：コンパイラを用いた情報フロー制御による情報漏洩防止機構，研究報告コンピュータセキュリティ（CSEC），Vol. 2015, No. 13, pp. 1–8 (2015).
[7] Portokalidis, G., Slowinska, A. and Bos, H.: Argos: An Emulator for Fingerprinting Zero-day Attacks for Advertised Honeypots with Automatic Signature Generation, *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, EuroSys '06, pp. 15–27 (2006).
[8] Bellard, F.: QEMU, a Fast and Portable Dynamic Translator, *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '05, pp. 41–41 (2005).
[9] Kawakoya, Y., Iwamura, M., Shioji, E. and Hariu, T.: API Chaser: Anti-analysis Resistant Malware Analyzer, *The 16th International Symposium on Research in Attacks*, pp. 23–25 (2013).
[10] Song, D., Brumley, D., Caballero, J., Jager, I., Kang, M. G., Liang, Z., Newsome, J., Poosankam, P. and Saxena, P.: BitBlaze: A new approach to computer security via binary analysis, *In Proceedings of the 4th International Conference on Information Systems Security*, pp. 1–25 (2008).
[11] Yin, H., Song, D., Egele, M., Kruegel, C. and Kirda, E.: Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis, *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pp. 116–127 (2007).
[12] Zhu, D. Y., Jung, J., Song, D., Kohno, T. and Wetherall, D.: TaintEraser: Protecting Sensitive Data Leaks Using Application-level Taint Tracking, *SIGOPS Operating Systems Review*, Vol. 45, No. 1, pp. 142–154 (2011).
[13] Luk, C.-K., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., Wallace, S., Reddi, V. J. and Hazelwood, K.: Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation, *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, pp. 190–200 (2005).
[14] 栗田弘之，塩谷亮太，入江英嗣，五島正裕，坂井修一：動的なインフォメーションフロー制御による情報漏洩防止手法，研究報告計算機アーキテクチャ（ARC），Vol. 2007, No. 17, pp. 227–232 (2007).
[15] Bochs: The Open Source IA-32 Emulation Project, `http://bochs.sourceforge.net/` (2015).
[16] Microsoft TechNet: Enterprise data protection (EDP) overview, `https://technet.microsoft.com/en-us/library/dn985838(v=vs.85).aspx` (2015).