情報処理学会論文誌 Vol.60 No.12 2269–2278 (Dec. 2019)

TCPによるネットワークワイドなテイント追跡を用いた情報漏洩防止システム

松本 隆志[1,†1] 瀧本 栄二[1] 齋藤 彰一[2] 毛利 公一[1,a)]

受付日 2019年3月10日, 採録日 2019年9月11日

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．我々はこの問題に対して，これまでファイルごとにデータ保護ポリシを設定可能（以下，ポリシ）とし，ポリシで禁止された範囲へのプロセスによるデータの出力・送信を検出・禁止することによって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 Salvia シリーズの開発を行ってきた．それぞれの差異は，主として，プロセスが扱うデータフローの追跡手法であった．本論文では，動的テイント解析機能を有するハードウェアエミュレータと OS の連携によって実現した TA-Salvia について述べる．TA-Salvia の特徴は，(1) 1 バイト単位でポリシを定義できる，(2) メモリ上において 1 バイト単位でデータフロー追跡が可能，(3) 二次記憶装置においても同様に追跡が可能，(4) TA-Salvia どうしであればネットワークを越えたデータフローの追跡も可能な点である．本論文では，特に (4) について，その設計，実装および評価について述べる．評価では，実際にファイル共有やメール送信を行い，データを継続して追跡できていることを示した．

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

Data Leakage Prevention System using Network-wide Dynamic Taint Tracking by TCP

Takashi Matsumoto[1,†1] Eiji Takimoto[1] Shoichi Saito[2] Koichi Mouri[1,a)]

Received: March 10, 2019, Accepted: September 11, 2019

Abstract: Many data leakage incidents are caused by human errors. To solve this problem, we have been developing the Salvia secure system series which prevent data leakage caused by such errors. They have a basic concept that users can define data protection policies for each file, and that outputting or sending activities of processes will be controlled based on the policies. The major difference among them is a technique for tracing data flow. In this paper, we describe TA-Salvia which is based on collaboration between CPU emulator with dynamic taint analysis and operating system. TA-Salvia has these features: (1) Data protection policies can be defined for each byte, (2) Each byte on memory can be traced as data flow, (3) Each byte on storage can be traced as data flow, and (4) Each byte over TCP between TA-Salvia hosts can be also traced as data flow. In this paper, we describe design and implementation of TA-Salvia, especially about (4). And we show that tracing data flow over TCP, and that they can be controlled by policies.

Keywords: data leakage prevention, file access control, dynamic taint analysis, operating system

[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
[†1] 現在，日本電気株式会社
Presently with NEC Corporation, Minato, Tokyo 108–8001, Japan
[a)] mouri@cs.ritsumei.ac.jp

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 2017 年情報セキュリティインシデントに関する調査報告書 [6] によると情報漏洩の発生件数の約 60%は，「管理ミス」，「誤操作」および「紛失・置忘れ」が原因と

© 2019 Information Processing Society of Japan

なっている．「管理ミス」は，業務上において，作業手順の誤りや情報の公開・管理のルールが明確化されていなかったことが原因のミスである．「誤操作」は，宛先の書き間違いや操作ボタンの押し間違いなどのことである．「紛失・置忘れ」は，持出し許可を得た情報を持出し先や移動中に忘れたり，紛失したりすることである．このように，情報漏洩のほとんどは，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ものである．

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，自由にデータを扱えるため，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 [7] や TOMOYO Linux [8] 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ごと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，多数のファイルについて，それぞれが異なる機密度が設定される場合などについては，すべてに適応したセキュリティポリシを記述することは難しい．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ほかに，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，我々は，Salvia Linux [1]，DF-Salvia [2]，User-mode DF-Salvia [3] 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，ファイルごとに設定可能とする．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せる場合には向かない．また，プログラムの静的解析において，変数のエイリアスの解析などにおいて限界がある．

以上の背景から，我々はこれまで，新たな Salvia システムとして TA-Salvia の開発を行ってきた．TA-Salvia は，Argos [9] と呼ばれる動的テイント解析機能を有するハードウェアエミュレータと OS が連携することで情報漏洩を防止するシステムである．具体的には，ユーザプロセスが扱うデータフローを Argos の動的テイント解析機能を用いて追跡し，その追跡結果に基づいて OS がデータの出力の可否を判定することで，情報漏洩を防止する．TA-Salvia の技術的特徴は，(1) 1 バイト単位でポリシを定義可能，(2) メモリ上において 1 バイト単位でデータフロー追跡が可能 [4]，(3) 二次記憶装置においても同様に追跡可能 [5]，(4) TA-Salvia どうしであればネットワークを越えたデータフローも追跡可能な点である．このように，TA-Salvia は，物理メモリ上のすべてのデータを追跡・制御可能であるため，システム全体のアプリケーションに透過的に適用できるとともに，静的解析のエイリアス問題が発生しない．また，ポリシをファイルごとではなく，1 バイト単位で管理しているため，複数のファイルが 1 つのファイルに統合されたとしても，1 バイト単位でデータの出力を制御できるといった機能的特長を有している．

本論文では，特に (4) の TA-Salvia による保護範囲をネットワークにまで拡張する手法について述べる．これまでは，1 つの端末内のデータフローのみを対象に追跡を行ってきた．しかし，実際の環境では，複数の端末が協調して動作することが多い．こういった環境に対応するためには，ネットワークに送信された場合でも継続して，受信側でデータフローの追跡を行えるようにし，情報漏洩を未然に防ぐことが必要である．これを TA-Salvia で行うために，TCP パケットに対してテイント情報を付与することで実現した．本論文の貢献は，以下の 2 点である．

- ネットワークワイドで情報漏洩を防止するシステムである TA-Salvia を提案した．
- 実装と実験による評価を通して，目的とする機能の実現可能性を明らかにした．

以下，本論文では，2 章で TA-Salvia の基本構成について述べ，3 章で提案手法について述べる．4 章で実装について述べ，5 章で評価について述べる．6 章で関連研究について述べ，7 章でまとめる．

2. TA-Salvia の基本構成

TA-Salvia は，ユーザプロセスが扱うデータフローを Argos の動的テイント解析機能を用いて追跡し，その追跡結果に基づいて OS がデータの出力の可否を判定することで，情報漏洩を防止する．TA-Salvia の全体構成を**図 1** に示す．本章では，TA-Salvia における Argos の動的テイント解析機能，シャドウ領域制御用のインタフェース，ポリシの管理方法，バイト単位のアクセス制御について述べ，最後に全体の動作の流れについて述べる．

© 2019 Information Processing Society of Japan

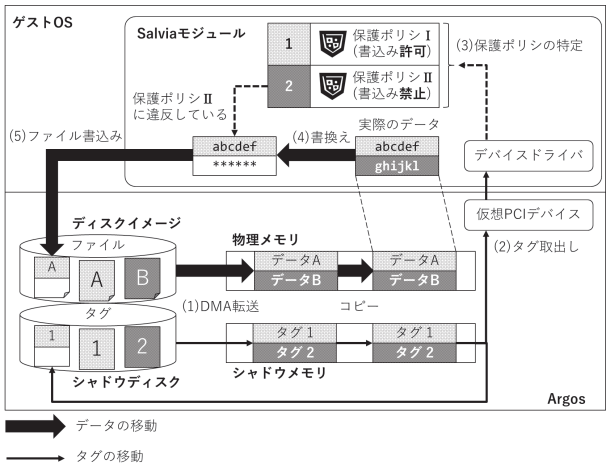

図 1 TA-Salvia の全体構成
Fig. 1 Overview of TA-Salvia.

2.1 Argos の動的テイント解析機能

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ることで，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．データに設定され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．

TA-Salvia が利用している Argos は，QEMU [10] と呼ばれるハードウェアエミュレータに動的テイント解析機能を追加したシステムである．動的テイント解析を実現するためには，データにタグを設定できるような仕組みが必要である．Argos は，データにタグを設定するために，物理メモリとレジスタに 1 対 1 に対応するタグ保持用のシャドウメモリとシャドウレジスタを用意している．Argos は，仮想 CPU が命令をフェッチするたびにタグを確認し，Argos のタグ伝播ルールに基づいてタグを伝播させることでデータの追跡を可能にしている．TA-Salvia では，この Argos をさらに拡張し，ディスク上のデータに対してもタグを保持できるようにしている．具体的には，エミュレータで使用するディスクイメージと同じサイズのタグ保持用のシャドウディスクイメージ（以下，シャドウディスク）を用意し，シャドウディスクとシャドウメモリ間でタグを伝播できるように拡張した．これにより，メモリだけでなくディスク上のファイルのデータも追跡することが可能となる．

2.2 シャドウ領域制御用インタフェース

TA-Salvia は，Argos の動的テイント解析に使用されるタグを OS が活用することでアクセス制御を行う．しかし，Argos は，OS にタグを付与・取得させるための機能を持たない．そのため，TA-Salvia は，OS からシャドウ領域を制御するためのインタフェースとして，Argos に仮想 PCI デバイスを追加した．また，OS には，そのデバイスを利用するためのデバイスドライバを用意した．OS は，デバイスドライバを通して Argos の仮想 PCI デバイスにコマンドを送信する．Argos は，仮想 PCI デバイスを通して OS のデバイスドライバにコマンドの実行結果を返す．このような仕組みで OS は，シャドウメモリにタグ付け・取出しなどを行っている．

2.3 保護ポリシの管理

TA-Salvia は，ポリシを YAML 形式で記述することができる．ポリシは，制御方法，ユーザ，グループの指定など詳細に記述可能である．この YAML 形式のポリシをパースしたものをファイルとして/etc/salvia 以下に保存しておく．TA-Salvia は，起動時に自動的に/etc/salvia 以下のファイルを読み出し，ポリシ管理テーブルに登録する．また，起動後にポリシを変更する場合は，専用のシステムコールを呼び出すことで行える．ポリシ管理テーブルに登録されたポリシは，タグと紐付けられ，アクセス制御時にタグから参照される．

2.4 アクセス制御方法

TA-Salvia は，情報漏洩の要因となるシステムコールをフックし，アクセス制御機能を追加している．TA-Salvia は，制御したいデータを伏せ字（アスタリスク）に置き換えることで，バイト単位のアクセス制御を実現している．たとえば，1 番のタグに「ファイルへの書込みを禁止する」というポリシが紐付けられていたとする．このとき，TA-Salvia は，1 番のタグと対応するデータをすべてアスタリスクに変換することで，出力を禁止する．

2.5 全体の動作

ポリシの作成やファイルへのタグ付けは，実際のファイルアクセスよりも前に設定しておく．ファイルアクセス時の TA-Salvia の動作の流れは，以下のようになる．

(1) ディスク上のファイルは，OS の先読み処理により，ディスクキャッシュとしてメモリに転送される．この転送と同時に Argos は，シャドウディスク上の対応するタグをシャドウメモリに伝播させる．
(2) ファイル書込み時に OS は，書き込むデータと対応するタグを Argos のシャドウメモリから取得する．タグ取出しは，仮想 PCI デバイスとそれを制御するデバイスドライバを通して行われる．
(3) 取得したタグとポリシ管理テーブルを用いて，適用すべきポリシを特定する．
(4) ポリシに違反したデータがあれば，そのデータをアスタリスクに置き換える．アスタリスクに置き換えたデータのタグは，0（タグなし）に変更される．
(5) ファイル書込みにより，ファイルのデータがメモリからディスクに転送される．この転送と同時に Argos は，シャドウメモリ上の対応するタグをシャドウディ

© 2019 Information Processing Society of Japan

スクに伝播させる．

3. 提案手法

本章では，まず TA-Salvia が目指す環境のモデルについて述べる．次に，そのモデルを実現するために必要なタグとポリシの共有方法について述べる．

3.1 TA-Salvia が目指す環境のモデル

TA-Salvia は，これまで 1 つの端末内のデータフローのみを対象に追跡してきた．しかし，実際の環境では，複数の端末が協調して動作することが多い．たとえば，ファイル共有ソフトである Samba や NFS，またメールの送受信を管理する Postfix や Sendmail などは，複数の端末間でデータが送受信される．このようなアプリケーションは，情報漏洩の要因となりやすい．TA-Salvia は，上記のような環境を想定し，**図 2** のようなモデルを目指している．情報漏洩の要因となりやすいアプリケーションを TA-Salvia 内で動作させる．サーバとクライアントに分ける場合は，別々の TA-Salvia を稼働させ，それぞれが協調して情報漏洩を防止する．これを実現するためには，ネットワークに送信された場合でも継続してデータの追跡を行えるような仕組みが必要である．

3.2 タグとポリシの共有機能

送信されたデータを追跡するためには，データに対応するタグとポリシを各 TA-Salvia が共有する必要がある．ポリシの共有は，各 TA-Salvia 上でポリシ同期用のデーモンを稼働させておくことで実現する．タグの共有は，Netfilter を用いて実装する．タグ共有機能の概要図を**図 3** に示す．Netfilter は，パケットのフィルタリングや NAT 機能を実現するために用いられるフレームワークである．送信されるパケットを Netfilter でフックし，パケットの末尾にデータと対応するタグを付与する．また，受信したパケットを Netfilter でフックし，パケットの末尾からタグを取り出す．このようにデータと同時にタグも送信することで，タグの共有機能を実現する．タグの共有手順は，以下のとおりに行われる．

(1) コネクション確立前に通信相手が TA-Salvia かどうかを判定する．
(2) 送信相手が TA-Salvia であれば，データと対応するタグをシャドウメモリから取り出す．
(3) 取り出したタグをパケットの末尾に付与して送信する．
(4) 受信側は，パケットの末尾からタグを取り出す．
(5) 取り出したタグをデータと対応するシャドウメモリの領域にタグ付けする．

なお，通常ファイルは，パケットロスによってデータが消失しない信頼性の高い通信として TCP 通信が用いられる．このことから，TCP 通信に焦点を当てた実装を行った．

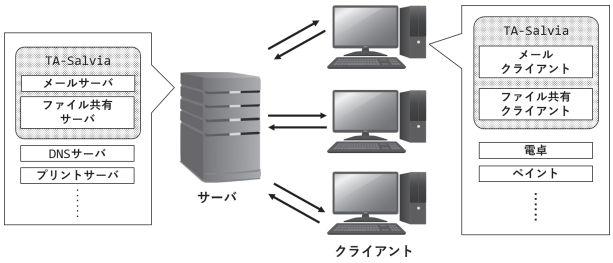

図 2 TA-Salvia が目標とする環境のモデル
Fig. 2 Targeted environment model by TA-Salvia.

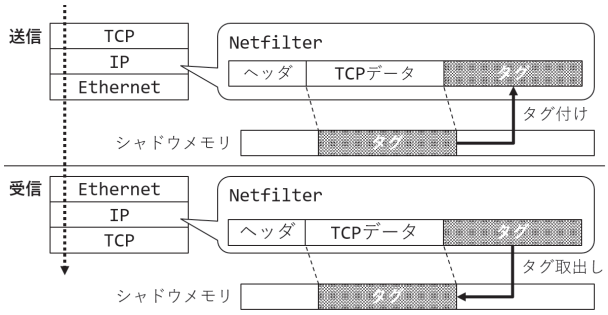

図 3 タグ共有機能の概要
Fig. 3 Overview of sharing tag between TA-Salvias.

4. 実装

本章では，タグの共有機能に必要な通信相手の特定，タグ用の領域確保，タグの有無の判定，パケットのタグ付け・取出しについて述べる．最後に，NIC のオフロード機能による影響について述べる．

4.1 通信相手の特定

まずは，自身が TA-Salvia であることを通知する方法について述べる．TA-Salvia は，TCP パケット送信前にヘッダを確認し，SYN フラグが設定されていれば，自身が TA-Salvia であることを示すフラグ（SLV フラグ）をヘッダの予約領域に設定する．これにより，通信相手に自身が TA-Salvia であることを通知することができる．この方法は，TCP の拡張機能が使用可能か調べるために使われる一般的なものであり，妥当であると考えられる．IP アドレスベースでの TA-Salvia か否かの判定は，DHCP や NAT による IP アドレスの変更・変換への対応が困難であるため，コネクション単位で確認することとした．

次に，通信相手が TA-Salvia であったことを保持する方法について述べる．TA-Salvia は，TCP パケット受信後にヘッダを確認し，SLV フラグが設定されていれば，通信相手が TA-Salvia であると判定する．このとき，sock 構造体の sk_flags メンバに通信相手が TA-Salvia であることを示すフラグを設定する．sock 構造体は，コネクション確立から終了まで使用される構造体であるため，ここにフラグを設定することで，コネクション終了まで保持することが可

© 2019 Information Processing Society of Japan

能である．また，sk_flags メンバは，3 bit 分の予約領域があるため，ここにフラグを設定することが可能である．

これ以降は，sk_flags メンバを確認することで TA-Salvia 間通信であることを判別できる．ただし，sock 構造体へのアクセスは，TCP パケット送信時と受信時で異なる．送信時は，すでに TCP レイヤで sock 構造体のルックアップ処理が行われているため，そのままアクセスすることが可能である．しかし，受信時は，まだルックアップ処理が行われていないため，アクセス前に自身でルックアップする必要がある．sock 構造体のルックアップは，__inet_lookup_skb 関数を用いることで可能である．

4.2 タグ用の領域確保

TA-Salvia は，TCP パケットの末尾にタグを付与し，送信することでタグの共有を実現する．しかし，Netfilter 内でタグを付与しようとすると，ソケットバッファの上限を超えてしまい，skb_over_panic となってしまう．また，ソケットバッファの上限を増やしてタグを付与したとしても，MTU（Maximum Transmission Unit）を超えてしまい TCP パケットは分割されてしまう．TCP パケットが分割されると，タグとデータの対応付けが行えない．そこで，MSS（Maximum Segment Size）を本来のサイズの半分に調整することでタグ用の領域を確保する手法を採用した．TCP は，MSS を基にセグメントサイズを決定している．そのため，MSS を調整することで，TCP パケットの末尾にタグを付与したとしてもソケットバッファの上限内に収まり，MTU を超えることもない．

MSS 調整前後のパケットの構成を**図 4** に示す．MSS は，スリーウェイハンドシェイク時に送信される SYN パケットのオプションに設定され，TCP は双方で最も小さい MSS を採用する．TA-Salvia は，この MSS オプションを半分のサイズに書き換える．TCP ヘッダを変更するため，チェックサムを更新する必要がある．

ここで，図 4 の TCP Data として “abcd” という文字列が入るものと仮定する．さらに，各文字にはタグとして 1，2，3，4 が付与されているものとする．このとき，TA-Salvia Tag 領域には “1234” というバイト列が入り，結果として “abcd1234” が TCP パケットのペイロードとなる．このため，TA-Salvia では，セグメントサイズの前半が実際のペイロード，後半がそれに対応するタグであると解釈して処理している．

上記の実装でほとんどの場合問題はないが，まれに受信バッファの許容量が MSS 以下となる場合がある．このとき，送信側の TCP は，ソケットバッファを切り詰めて確保してしまう．そのため，TCP パケットにタグを付与しようとすると，ソケットバッファの上限を超えてしまい，skb_over_panic となってしまう．この問題は，pskb_expand_head 関数を用いてソケットバッファを再度割り当てることで解決できる．

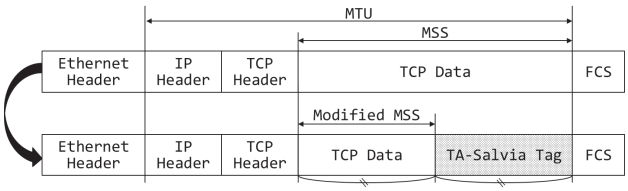

図 4 MSS 調整前後のパケットの構成
Fig. 4 Packet structure before and after adjusting MSS.

4.3 タグの有無の判定

TCP で送信されるすべてのデータにタグが付与されているとは限らない．TA-Salvia は，送信するデータにタグが付与されていない場合，タグが付与されていないことを示すオプション（NOTAG オプション）を TCP ヘッダに追加する．受信側の TA-Salvia は，NOTAG オプションを確認することでタグがないことを判断する．

NOTAG オプションは，オプション番号（0x23）とサイズ（0x02）の合計 2 バイトである．このオプションの挿入は，TCP パケット送信時に Netfilter 内で行う．このオプションの挿入方法について述べる．TCP ヘッダのオプション領域は，可変長であり総サイズが 4 バイトの倍数になるように調整される．領域の調整には，1 バイトの No-Operation（NOP）オプションが用いられる．すでに 2 つ以上の連続した NOP オプションが設定されている場合，それらを NOTAG オプションに置き換える．そうでない場合は，TCP ヘッダの末尾に NOTAG オプションと 2 つの NOP オプションを挿入する．このとき，TCP ヘッダのサイズが変更されるため，TCP ヘッダのデータオフセットとチェックサムを更新する必要がある．また，全体のパケットサイズも変更されるため，IP ヘッダのパケット全長とチェックサムも更新する必要がある．

4.4 パケットのタグ付け・取出し

TCP パケットのタグ付け手順（**図 5**）について述べる．送信側の TA-Salvia は，まず TCP パケットのデータにタグが付与されているかを確認する．タグが付与されていた場合，ソケットバッファを TCP のデータサイズ分拡張する．ソケットバッファの拡張は，skb_put 関数を用いることで可能である．次に，データと対応するタグを Argos のシャドウメモリから取得し，拡張された領域にそのタグを設定する．このとき，パケットサイズが変更されるため，IP ヘッダのパケット全長とチェックサムを更新する．

TCP パケットのタグ取出し手順（**図 6**）について述べる．受信側の TA-Salvia は，まず NOTAG オプションが設定されているかを確認する．NOTAG オプションが設定されていない場合，TCP パケットの末尾にタグが付与されている．このとき，パケットの末尾からタグを取り出し，

© 2019 Information Processing Society of Japan

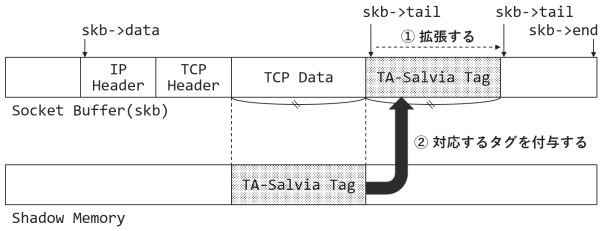

図 5 TCP パケットのタグ付け
Fig. 5 Storing tag information into TCP packet.

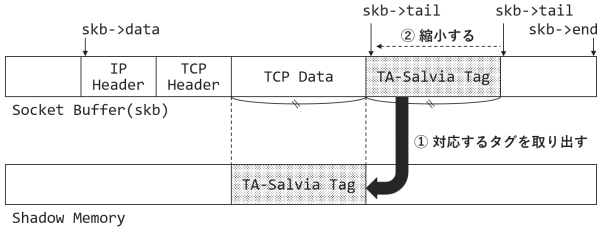

図 6 TCP パケットのタグ取出し
Fig. 6 Extracting tag information from TCP packet.

データと対応するシャドウメモリ領域にそのタグを設定する．タグ取出し後，取り出したタグサイズ分ソケットバッファを縮小する．ソケットバッファの縮小は，skb_trim 関数を用いることで可能である．このとき，パケットサイズが変更されているが，受信時はすでに IP レイヤを通過しているため，IP ヘッダのパケット全長やチェックサムを更新する必要はない．また，4.2 節で述べた形式でデータとタグが連続して格納されているため，タグの位置や対応に関する情報を必要とせずにそれらを適切に分離できる．

4.5 オフロード機能の無効化

NIC には，オフロード機能がいくつかあり，中でも下記の 4 つが有効の場合，タグの送受信が正しく行われない可能性がある．

- TSO（Tcp Segmentation Offload）
- GSO（Generic Segmentation Offload）
- GRO（Generic Receive Offload）
- SG（Scatter/Gather）

まずは，TSO と GSO について述べる．TSO は，TCP のデータを NIC 内で MTU のサイズに分割する機能である．GSO は，送信するパケットのすべてが対象で，同様に MTU のサイズに分割する機能である．これらが有効になっている場合，TCP は，MSS のとおりに調整を行わず，ソケットバッファの上限までデータを格納する．そのため，TA-Salvia は，タグ用の領域を確保することができない．また，ソケットバッファの上限を増やしてタグを付与したとしても，NIC でパケットが分割されてしまうため，受信側の TA-Salvia で正しくタグ取出しを行えない．

次に，GRO について述べる．GRO は，受信したパケットのすべてが対象で，複数のパケットを NIC 内で結合する．これが有効になっている場合，複数のパケットが結合されてしまい，データとタグの順序が崩れてしまう．そのため，TA-Salvia は，パケットから正しくタグ取出しを行えない．

最後に SG について述べる．SG は，パケットを非連続な領域に分割して保持するための機能である．これが有効になっている場合，Netfilter でフックした時点では，ソケットバッファ内に TCP のデータが存在せず，別の領域に分割されて管理される．そのため，TA-Salvia は，Argos のシャドウメモリからタグを取得することができない．また，タグを取得できたとしても，複数の領域にパケットが分割されてしまうため，データと対応付けてタグを付与することができない．

5. 評価

本章では，TA-Salvia の機能評価と性能評価について述べる．機能評価では，ファイル共有とメールを用いて，ネットワークに送信されたデータが正しく追跡され，制御できるかどうかを確認した．また，性能評価では，FTP を用いてファイル転送を行い，本手法の実装による影響を確認した．

5.1 機能評価

ファイル共有とメールを用いた機能評価について述べる．共有されたファイルをメールに添付して送信する．添付されたファイルデータの出力を制御できるかを確認する．

5.1.1 評価環境

用意した環境を**図 7** に示す．サーバとクライアントの 2 つの TA-Salvia を用いる．まずは，サーバとして利用する TA-Salvia の構成について述べる．この TA-Salvia では，Postfix と Samba の 2 つサービスを稼働させている．また，secret.txt というファイルを Samba の共有ディレクトリに用意した．このファイルには，データの先頭から 12 バイト分に 1 番のタグを設定している．さらに，1 番のタグに「root ユーザは，すべてのアクセスを許可し，postfix グループは外部への送信を禁止する」というポリシを紐付けている．次に，クライアントとして利用する TA-Salvia の構成について述べる．この TA-Salvia には，メールクライアントの Sylpheed がインストールされている．また，Samba で共有されたディレクトリをマウントしている．ポリシは，サーバと同様のものが共有されている．以上の構成で，クライアントの TA-Salvia が secret.txt をメールに添付し，サーバの TA-Salvia の Postfix 経由で Gmail に送信する．

5.1.2 評価結果

メールのソースの一部を**図 8** と**図 9** に示す．図 8 は，Postfix が Gmail に送信する前のメールで，図 9 は Gmail が受信したメールである．6 行目は，添付されたファイル

© 2019 Information Processing Society of Japan

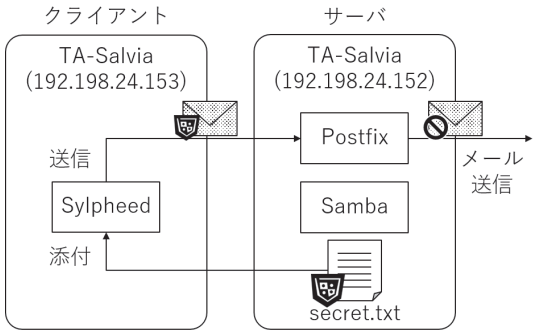

図 7 ファイル共有とメールを用いた評価環境
Fig. 7 Evaluation environment.

```
1 --Multipart=_Fri__8_Sep_2017_02_41_43_+0000_Qhg=AvAZv7FTrH1F
2 Content-Type: text/plain; name="secret.txt“
3 Content-Disposition: attachment; filename="secret.txt“
4 Content-Transfer-Encoding: 7bit
5
6 SECRET DATA
7
8 --Multipart=_Fri__8_Sep_2017_02_41_43_+0000_Qhg=AvAZv7FTrH1F--
```

図 8 メールのソースの一部（Postfix が送信する前のメール）
Fig. 8 Part of mail before being sent by Postfix.

```
1 --Multipart=_Fri__8_Sep_2017_02_41_43_+0000_Qhg=AvAZv7FTrH1F
2 Content-Type: text/plain; name="secret.txt“
3 Content-Disposition: attachment; filename="secret.txt“
4 Content-Transfer-Encoding: 7bit
5
6 **********
7
8 --Multipart=_Fri__8_Sep_2017_02_41_43_+0000_Qhg=AvAZv7FTrH1F--
```

図 9 メールのソースの一部（Gmail が受信したメール）
Fig. 9 Part of mail received by Gmail.

（secret.txt）のデータである．Gmail が受信したメールだけがアスタリスクに変換されていることから，「postfix グループは，外部への送信を禁止する」というポリシに従い，出力が禁止されていることが分かる．また，アスタリスクに変換されたデータが 6 行目だけであることからバイト単位の制御も行えていることが分かる．さらに，ファイル共有された secret.txt のデータに対してアクセス制御が行えたことから，ネットワークに送信されたとしても継続してデータの追跡が行えていることも確認できる．

5.2 性能評価

FTP を用いた性能評価を行った．FTP を用いて TA-Salvia 間でファイル送信を行い，ファイル転送が完了するまでの時間を計測した．

5.2.1 評価環境

評価のために，ランダムなデータが書き込まれた 1 MiB ファイルを用意した．また，I/O による影響をなくすために，tmpfs 上にファイルを用意している．比較対象は，以下の 5 つである．

(1) Original Kernel 3.9.9 + QEMU 1.1.0
(2) Original Kernel 3.9.9 + Argos 0.7.0（MTU：1,500 byte）
(3) Original Kernel 3.9.9 + Argos 0.7.0（MTU：770 byte）
(4) TA-Salvia（データにタグなし）
(5) TA-Salvia（データにタグあり）

Original Kernel 3.9.9 は，何も変更を加えていないカーネルのことである．バージョンは，TA-Salvia がベースとしているものを使用した．同様に，QEMU 1.1.0 と Argos 0.7.0 は，何も変更を加えておらず，バージョンは Argos と TA-Salvia がベースとしているものを使用した．(3) では，MTU を 770 byte に調整している．これは，TA-Salvia が MSS を半分に調整した状態と同じ条件にするためである．(4) では，データにタグを設定していないため，TA-Salvia は，パケットのタグ付け・取出しとアクセス制御を行わない．(5) では，送信するファイルのデータすべてに 1 番のタグを設定している．また，TA-Salvia には，あらかじめ 1 番のタグに「すべてのアクセスを許可する」というポリシを紐付けている．タグを設定しているため，TA-Salvia は，パケットのタグ付け・取出しとアクセス制御を行う．

5.2.2 評価結果

本項では，FTP によるファイル転送時間の評価結果とそれぞれの環境の比較・考察について述べる．評価結果を**図 10** に示す．

まずは，(1) と (2) の比較について述べる．(1) と (2) の違いは，Argos による動的テイント解析機能の有無である．評価結果では，Argos を使用した場合に転送時間が約 5.2 倍増加することが確認できた．これは，動的テイント解析によるオーバヘッドが原因である．(1) 以外は，すべての処理に対して同程度の動的テイント解析のオーバヘッドがかかってしまうことが予想される．

次に，(2) と (3) の比較について述べる．(2) と (3) の違いは，MTU のサイズである．MTU がデフォルト値の 1,500 byte より小さい 770 byte であるため，パケット数が 2 倍近く増加する．増加したパケット数だけ，TCP/IP の処理が増加してしまうため，全体的な転送時間が長くなってしまう．このことから，TA-Salvia がタグ用の領域を確保のために MSS を半分に調整すると，パケット数が増加し，その影響で転送時間が約 2.3 倍増加してしまうことが分かる．

さらに，(3) と (4) の比較について述べる．(3) と (4) の違いは，Netfilter の機能の有無である．TA-Salvia は，データにタグが付与されていない場合，NOTAG オプションの挿入とその確認しか行っていない．したがって，(3) と (4) の転送時間の差は，Netfilter を追加したことが主な原因である考えられる．ただし，Argos は，Netfilter の基本的な処理すべてに対しても，動的テイント解析を行っているため，これが影響している可能性がある．

最後に，(4) と (5) の比較について述べる．(4) と (5) の違いは，データにタグが付与されているかどうかである．

© 2019 Information Processing Society of Japan

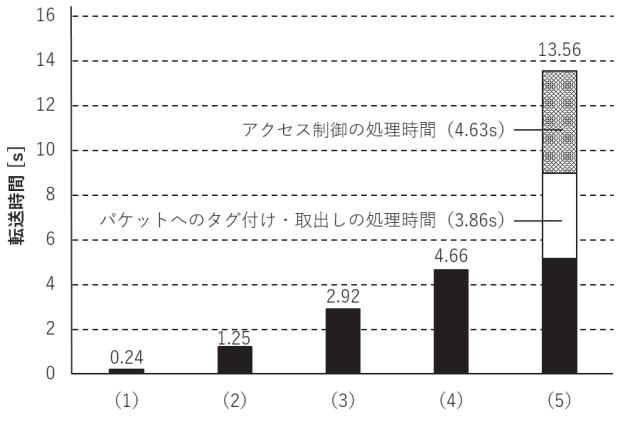

図 10 FTP によるファイル転送時間
Fig. 10 File transfer time by FTP.

すなわち，それぞれ TA-Salvia の通信処理の最良値（全パケットにタグ領域が含まれていない場合）と最大値（全パケットにタグ領域が含まれる場合）を表している．実際の運用では，タグが付与されていないパケットとタグが付与されたパケットが混在することが考えられる．その場合，それらの割合に応じて (4) と (5) の範囲内の性能になる．TA-Salvia は，データにタグが付与されていた場合，TCP パケットにタグ付けと取出しを行う．また，FTP でファイルを転送するとき，FTP サーバとクライアントは，対象となるファイルを送信・保存するために write システムコールを発行する．TA-Salvia は，タグの付与されたデータが書き込まれるときにポリシに基づいたアクセス制御を行う．これらの処理により，転送時間が約 8.49 秒増加してしまっている．この転送時間の増加の主な原因は，シャドウ領域制御用インタフェースによる Argos とゲスト OS 間の通信である．この通信によるオーバヘッドは，送受信されるデータサイズに依存する．TA-Salvia でタグ付きのデータを送信する際は，MSS が小さくなるため送信パケット数がタグのついてないデータを送信するよりも，同じデータサイズであってもパケット数がおおむね 2 倍に増加する．この場合，パケット数が増加することによるオーバヘッドは増加するが，Argos とゲスト OS 間の通信によるオーバヘッドに変化はない．この処理は，次節で述べるようにまだチューニングの余地があり，想定では (4) と同等にまで削減できると考えている．

5.3 考察と検討課題

本節では，TA-Salvia の限界および性能改善に関して，考察と今後の検討課題について述べる．

図 10 に示したように，TA-Salvia による通信時間は最小で図 10 (4)，最大で図 10 (5) と大きく増加する．そのため，多数のクライアントが接続されるクライアントサーバシステムにおいてスケーラビリティに欠ける．上述のとおり，その大きな要因は Argos とゲスト OS 間の通信である．シャドウ領域制御用インタフェースは仮想 PCI デバイスとして実現されており，その操作には IN 命令と OUT 命令が用いられる．1 バイトを単位とした入出力であるため，Argos と OS 間の通信時間が大きくなる．その改善として，シャドウ領域制御用インタフェースにメモリ領域を単位として通信するための命令を追加し，IN/OUT 命令の発行回数を減らすことが考えられる．または，Argos が管理するシャドウ領域を OS が直接参照できるよう，シャドウ領域を OS のメモリ空間にマッピングすることが考えられる．特に，後者の方法は Argos と OS 間の通信を必要としないため，大きな性能改善が期待できる．ただし，OS のメモリモデルと実装のモデルを大幅に変更する必要がある．また，ペイロードの半分をタグが占めるため，タグの圧縮などによる効率化も検討する必要がある．

TA-Salvia はデータ保護を主目的としており，本論文ではデータ保護を実現するための手法と実装に基づく実現可能性について述べている．実用性という面からは，大きなオーバヘッドという課題が残るものの，性能を犠牲にしてでもデータ保護を確実に行いたいという要求があれば，十分に実用的な機能を備えていると考えることができる．ただし，中間者攻撃やなりすましによるタグの削除，書換えなどが行われるとシステム全体が混乱する問題を抱えている．これについては，既存の暗号化・認証技術を応用することで攻撃を防ぐことが可能であると考える．

このように，TA-Salvia は実用に足る機能を備えてはいるものの，オーバヘッド削減や攻撃への耐性といった課題が残っている．今後は，これらの課題を解決し，完成度を高めていく必要がある．

6. 関連研究

データフローを追跡して情報漏洩の検知や防止を行っている研究は，数多く存在する．なかでも，本研究に類似した研究として TaintEraser [11] がある．TaintEraser は，Intel が開発した Pin [12] 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．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ときに，ランダムなデータに置き換えることで情報漏洩の防止を実現している．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

© 2019 Information Processing Society of Japan

ため，システム全体のアプリケーションに共通に適用させる場合には向かない．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．また，TaintEraser は，プロセス間通信を考慮していないため，データが送信されたとき，テイントを伝播できず漏洩してしまう可能性がある．TA-Salvia は，共有メモリやソケットを用いた場合においても追跡が可能である．さらに，本手法により，ネットワークに送信された場合でも継続して追跡が可能である．

TaintEraser のような通信時にデータの追跡が途切れてしまうという問題を解決した TaintExchange [13] という研究がある．TaintExchange は，動的テイント解析フレームワークの libdft [14] で構成されたツールである．libdft は，TaintEraser と同様に Pin を用いて動的テイント解析機能を実現している．TaintExchange は，libdft の基本的な動的テイント解析機能に加え，クロスプロセス・クロスホストでも継続してデータの追跡が行える．TaintExchange は，追跡したデータがソケットやパイプなどを用いて送信されるときに，ヘッダとしてテイント情報をデータに付与して送信する．このヘッダ情報を利用することで，テイント伝播を実現している．TaintExchange は，ビットマップを採用しているため，複数のデータを識別できない．一方，TA-Salvia は，バイトマップを採用しているため，複数のデータを個々に識別することが可能である．また，TaintExchange は，通信相手の特定方法について本文中で述べていない．テイント解析機能を持たないプロセスに対してデータが送信された場合，正しく通信できない可能性がある．TA-Salvia は，コネクション確立前に通信相手が TA-Salvia であることを識別している．

7. おわりに

本論文では，TA-Salvia による保護範囲をネットワークにまで拡張する手法について述べた．この手法を実現するために，データの追跡に必要なタグとアクセス制御に必要なポリシの共有機能を実装した．ポリシの共有は，各 TA-Salvia 上でポリシ同期用のデーモンを稼働させておくことで実現した．また，タグの共有は，Netfilter を用いて TCP パケットの末尾にタグを付与することで実現した．実際に，ファイル共有やメール送信が行われる環境で機能評価を行い，ネットワークに送信されたとしても継続して追跡・制御が行えることを確認した．さらに，FTP を用いた性能評価を行った．何も変更を加えていない環境に比べて，約 3.7〜10.8 倍転送時間が増加してしまうことを確認した．

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，Vol.47, No.SIG3 (ACS13), pp.1–15 (2006).

[2] 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).

[3] 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，情報処理学会研究報告，Vol.2015-CSEC-68, No.13, pp.1–8 (2015).

[4] 松本隆志, 大月勇人, 明田修平, 齋藤彰一, 毛利公一：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，情報処理学会研究報告，Vol.2016-CSEC-72, No.33, pp.1–8 (2016).

[5] 松本隆志，明田修平，瀧本栄二，齋藤彰一，毛利公一：動的テイント解析機能を利用した OS による細粒度データ出力制御手法，情報処理学会研究報告，Vol.2016-CSEC-75, No.1, pp.1–8 (2016).

[6] NPO 日本ネットワークセキュリティ協会：2017 年情報セキュリティインシデントに関する調査報告書【速報版】，入手先 ⟨https://www.jnsa.org/result/incident/⟩ (2018).

[7] Loscocco, P. and Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System, *Proc. FREENIX Track: 2001 USENIX Annual Technical Conference*, pp.29–42 (2001).

[8] 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，*Proc. Linux Conference 2004*, pp.1–10 (2004).

[9] Portokalidis, G., Slowinska, A. and Bos, H.: Argos: An Emulator for Fingerprinting Zero-day Attacks for Advertised Honeypots with Automatic Signature Generation, *Proc. 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, *EuroSys '06*, pp.15–27 (2006).

[10] Bellard, F.: QEMU, a Fast and Portable Dynamic Translator, *Proc. Annual Conference on USENIX Annual Technical Conference*, *ATEC '05*, p.41 (2005).

[11] 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).

[12] 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, *Proc. 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, *PLDI '05*, pp.190–200 (2005).

[13] Zavou, A., Portokalidis, G. and Keromytis, A.D.: Taintexchange: A Generic System for Cross-process and Cross-host Taint Tracking, *Proc. 6th International Conference on Advances in Information and Computer Security*, *IWSEC'11*, pp.113–128 (2011).

[14] Kemerlis, V.P., Portokalidis, G., Jee, K. and Keromytis, A.D.: Libdft: Practical Dynamic Data Flow Tracking for Commodity Systems, *SIGPLAN Not.*, Vol.47, No.7, pp.121–132 (2012).

© 2019 Information Processing Society of Japan

松本 隆志

2016 年立命館大学情報理工学部情報システム学科卒業，2018 年同大学大学院情報理工学研究科博士課程前期課程修了．同年日本電気株式会社入社，サイバーセキュリティ戦略本部へ配属，2019 年国立研究開発法人情報通信研究機構サイバーセキュリティ研究所サイバーセキュリティ研究室へ出向，現在に至る．主にシステムソフトウェア，サイバーセキュリティに関する研究に従事．

瀧本 栄二 （正会員）

1999 年立命館大学理工学部情報学科卒業，2001 年同大学大学院理工学研究科博士前期課程修了，2005 年同研究科博士後期課程単位取得退学，同年（株）ATR 適応コミュニケーション研究所専任研究員，2010 年立命館大学情報理工学部情報システム学科助手，2017 年立命館大学情報理工学部情報理工学科助教，現在に至る．主にシステムソフトウェア，無線ネットワークに関する研究に従事．博士（工学）．電子情報通信学会会員．

齋藤 彰一 （正会員）

1993 年立命館大学理工学部情報工学科卒業．1995 年同大学大学院博士前期課程修了．1998 年同大学大学院博士後期課程単位習得中退．同年和歌山大学システム工学部情報通信システム学科助手．2003 年同講師，2005 年同助教授．2006 年名古屋工業大学大学院助教授，2007 年同准教授，2016 年同教授．現在に至る．オペレーティングシステム，インターネット，セキュリティ等の研究に従事．博士（工学），ACM，IEEE-CS 各会員．

毛利 公一 （正会員）

1994 年立命館大学理工学部情報工学科卒業，1996 年同大学大学院理工学研究科修士課程情報システム学専攻修了，1999 年同研究科博士課程後期課程総合理工学専攻修了．同年東京農工大学工学部情報コミュニケーション工学科助手，2002 年立命館大学理工学部情報学科講師，2004 年同大学情報理工学部情報システム学科講師，2008 年同准教授，2014 年同教授となり，現在に至る．博士（工学）．オペレーティングシステム，仮想化技術，コンピュータセキュリティ等の研究に従事．電子情報通信学会，ACM，IEEE-CS，USENIX 各会員．

© 2019 Information Processing Society of Japan