汎用的利用シーンにおけるファイル追跡技術の提案

張 一凡 秋葉 淳哉 松村 隆宏 元田 敏浩

†NTT セキュアプラットフォーム研究所
180-8585 東京都武蔵野市緑町 3-9-11
{tyou.iifan, akiba.junya, matsumura.takahiro, motoda.toshihiro}@lab.ntt.co.jp

あらまし 本稿では，近年被害が増加する情報漏洩を抑止するために，コンテンツが能動的に自身への操作を監視，イベント発信を行う手法を用いて，情報漏洩を検知可能にするトレース技術を提案する．従来技術ではファイル操作の追跡のためには，操作環境に特化した監視ツールの導入しログを収集する必要があった．しかし，異なる組織間でコンテンツが流通する利用シーンを考えると漏洩した電子ファイルを開く環境にこのような監視ツールが導入されているとは期待できず，ファイルの操作追跡は困難であった．本稿では漏洩ファイルを操作する PC 環境依存性を解決するため，テイント解析技術を応用し，ファイルとして配布できるコンテナとそのコンテナ内のコンテンツに対する任意の処理をトレースする追跡技術を提案する．

A technology for tracing general file operations

Iifan Tyou, Junya Akiba, Takahiro Matsumura, Toshihiro Motoda †

†NTT Secure Platform Laboratories
3-9-11， Midori-cho Musashino-shi, Tokyo 180-8585

Abstract This paper proposes a solution which can detect leakage and trace derivational data of any digital files. In conventional method, we use a pre-installed monitor application for logging file operations on a PC. But we can't expect such an application is installed on the system of all PC environment which has the leaked file. In addition, to trace a detail operation done to a single file of general application is difficult at the moment. Our solution will solve this problem with the technology of taint analysis and a type of executable file wrapping method.

1 背景

近年ネットワークやクラウドサービスの普及に伴い情報漏洩事件が後を絶たない．顧客情報など機密性の高い情報が漏洩する場合も多く，企業は漏洩に対する責任を強く追求されるようになった．

情報漏洩リスクへの認識は浸透しつつあるが，業務の効率化や利便性向上のため，機密情報を取り扱う端末をインターネット接続させざるを得ない状況にある．情報漏洩の経路は多種に及び，端末に対する攻撃，人為的ミス，意図的な操作などに加え，標的型攻撃と呼ばれるソーシャルエンジニアリングなどを駆使した特定の対象に有効で高度な攻撃も増加してきた．業務効率の確保と前述のような高度な攻撃に対する対策を同時に実現することは難しい．いくつかの製品，サービスも広まりつつあるが，

攻撃手法の高度化に伴い，対策コストも上昇し続けている．一方で，企業は情報システム統制に関する説明責任を問われる風潮も高まっているため，漏洩防止と同時に漏洩後の対策の重要性も注目を集めている．

漏洩そのものを防ぐ技術として，ファイアウォールやIDS（Intrusion Detection System），DLP(Data Loss Prevention)などが広く研究されている．ファイルなど情報源の漏洩が機密情報そのものの漏洩に繋がらなくする技術として暗号技術やIRM(Information Rights Management)なども研究されてきた．これらの技術によるソリューションもすでに多数提案され，成果を上げている．しかし，前述の高度な攻撃者はそれらの技術や利用法を調査し，回避する複雑な攻撃も逐次考案しており，完全に漏洩を防止することは難しい．例えば暗号技術は情報漏洩対策として強力なツールではあるが，それを使う環境でショルダーハッキング，キーロガーやソーシャルエンジニアリングなど外部に依存する脆弱性により，安全性が低下している．

漏洩を防ぐことが困難な場合，セキュリティ対策として漏洩後の影響を最小限に抑えることが課題となる．例えば，漏洩ファイルを許諾されていない利用者が閲覧，保有していないか，不特定多数の利用者が閲覧していないかなど利用者の特定が可能になれば影響の大きさを予測できる．更に，特定した保有者への削除要求が可能であれば，漏洩後の対策にも繋がると考えられる．そこで，本研究では漏洩した情報の追跡（トレース）により，情報漏洩先の特定や事後対策を目的に考察検討を行う．この時，漏洩する情報の形式としてファイル，システムメモリやパケット形式等が考えられるが，本研究ではファイル経由の情報漏洩に対象を絞る．本研究の提案する方式によりトレースを行った結果，漏洩ファイルの保有者の環境の一覧と漏洩後に受けた操作の特定を可能にしたい．

2章では，本研究の目的であるファイル漏洩後の振る舞いを追跡するために必要となる既存技術とその問題を整理し，3章では2章で整理した問題に対する対策への検討を行う．その後，4章では 3 章で検討した対策を実現する方式を提案方式として紹介し，5章で提案方式を拡張，実現する時点での課題を整理する．

2　既存技術と課題

ファイルの振る舞いを追跡するプロダクトとして，先述のDLPや「統合ログ管理」と呼ばれるジャンルの製品がある．これらの製品も参考に，ファイル漏洩後の振る舞いを追跡するために必要となる技術要素として図 1のログ生成技術，転送技術，収集技術，トレース処理を抽出できた．これらの技術を組み合わせたトレースの流れではまずログ生成で追跡対象ファイルを監視し，ファイル操作に基づいて生成したログを送信，収集する．収集によって追跡システムにログが蓄積され，このログをトレースプログラムで処理することによって一連の意味のある情報に加工することができる．この内，汎用的な利用シーンを考えると，ログ送信[13]，ログ収集[14]や可視化[15]は既存技術が有り，技術的に工夫できるところが少ないため，ログ生成処理とトレース処理の構成を重点に既存技術と課題の整理を行う．以下ではそれぞれの処理に対する既存技術と課題を紹介する．

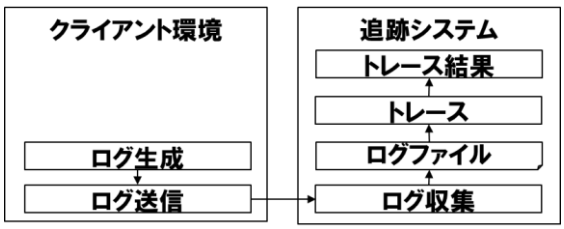

図 1ファイル追跡に必要な技術要素

2.1　ログ生成処理

ファイル操作に関するログ生成の際に重要となるのはユーザが実行したすべてのファイル操作に関するログを取得できることである．しかし，監視できる対象アプリケーションの多さ（汎用性）と操作の細かさ（精度）は監視方式に依存することがわかっている．本節では既存で広く用いられる監視方式としてOSドライバレイヤーでの監視方式と，アプリケーションレイヤーで

の監視方式を紹介し，本論文の目的である漏洩ファイルの操作を追跡するユースケースにおける有効性と課題を検討する．

2.1.1　OS ドライバレイヤーでの監視

追跡処理で有用なログを生成するための既存技術として，OS のドライバレイヤーに監視エージェントをインストールし，ファイル操作を監視し，ログを生成する手法(図 2)がある．このエージェントを用いることで，OS 上のすべてのファイル操作を追跡することができる．実装方式としては OS のドライバレイヤーでファイルシステムへのアクセスをフックする監視プログラムを用いてログを出力するものが一般的である．OS からファイルアクセスを行う際必ずファイルシステムのドライバを経由するので，連理的に OS 上の任意のプロセスによるファイル操作の操作ログを生成することが可能である．

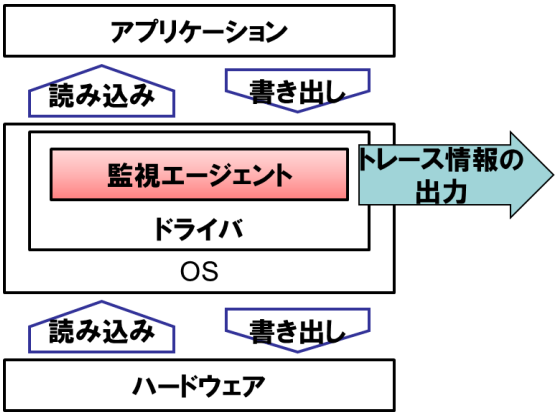

図 2 OS ドライバレイヤーでの監視

この方式の問題点は以下の通り整理できる．

汎用性問題：エージェントがインストールされていない環境でのトレースができない．組織内の統制が効く範囲においては，統制によりエージェントがすべてのクライアントにインストールされていると想定でき，その範囲内ではエージェントが生成したログからファイル操作をトレースできる．しかし，監視エージェントがインストールされてない環境での漏洩ファイルの操作はこの方式ではトレースできない．問題となる情報漏洩では特に，組織内統制が効かない箇所への漏洩が一般的であり，組織内での情報漏洩は少ない．漏洩ファイルが開かれるエージェントがインストールされていない端末でログを生成することが必要となる．

精度問題：この監視方式ではコピー&ペーストなどといったファイルシステムより上のアプリケーションレイヤーで実施される情報の行き来を監視できない．コピー&ペーストやメール送信，Cloud への保存など漏洩の経路は多様化しており，目的をファイルの漏洩に特化してもこれらの操作を無視することはできない．例えば，漏洩ファイルから部分的な情報をコピーし，別のファイルにペーストした場合，操作はアプリケーションレイヤーで実施され，ファイルシステムのドライバを監視するエージェント方式ではそのログを生成することができず，精度向上のための方式を検討する必要がある．

2.1.2　アプリケーションレイヤーでの監視

アプリケーションレイヤーでのファイル操作を監視する方式として，アプリケーションに対して監視エージェントを追加する方式(図 3)が考えられる．この方式の実装例として，OS が公開したフック API を利用する方式と，アプリケーションが IRM などのセキュリティ対策のために公開した API を用いて監視する方式が挙げられる．しかし，これらの API はユーザの処理に依存するもので，プログラム内処理として行われる転送等監視できない箇所もある．

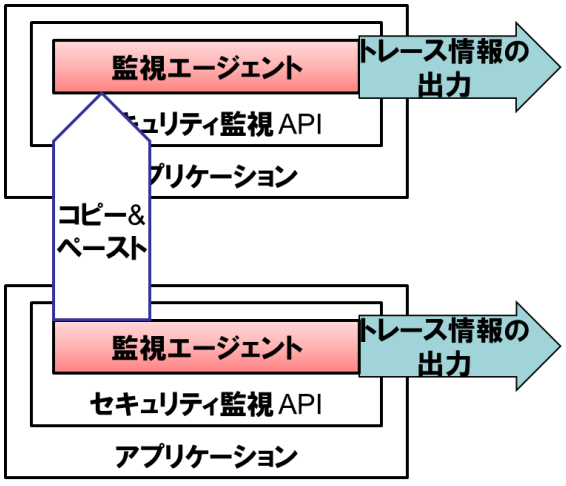

図 3 既存追跡情報の生成方式

この方式に対する問題は以下のものになる．

汎用性問題：2.1.1節の汎用性課題と同様エージェントが事前にインストールされていない環境ではログの生成ができない．更にこの方式では提供された API によって検知，監視できないアプリケーション処理に対してログを生成できない．このため，汎用的な利用シーンとして考えられる，任意のアプリケーションに対して，事前インストールなしで監視が可能になる技術の考案が必要になる．

2.2 トレース処理

トレース処理とは生成，収集された複数ログ間の関連性を追跡（トレース）し，ファイルに対して行われた操作を可視化するための処理である．既存のトレース処理方式としては DLP や資産管理，統合ログ管理などの製品では一般的にファイル操作のログを読みやすい型に変換し，つなぎ合わせる方式が一般的に用いられている．

利用シーンに合ったトレース処理がない場合，追跡に必要な情報（ログ）を正規化後，人間が検索し，一つ一つの操作の関連性を辿ることで一連のファイル操作の流れを追跡していた．人間が追跡を行う手法はソフトウェアのデバッグやシステムのエラー対処時に多く用いられており，検索を高速化するためのツールも提案されてきた．しかし，この手法では追跡情報に対する専門的知識や調べ方を理解したユーザにしか必要な情報を取り出せず，その処理も人間が行う場合非常に多くの時間を要する．そのため，追跡情報ログの出力製品に詳しくない利用者に対しては，トレース処理が必要となる．トレース処理では追跡情報を繋ぎあわせ，内包されている意味をわかりやすく表示することが求められる．本研究の目的である漏洩ファイルに対する操作の可視化の内容としてファイル操作の追跡では情報漏洩の有無や漏洩先を確認するため，漏洩先の特定に繋がる情報の図示が有効と考えられる．一つのファイルに着目するとその派生関係ではコピー等の操作によって時間の経過に伴いツリー状に広がる．このため樹形図で可視化することにより，樹形図上で漏洩ポイントとその影響を受ける派生ファイルの一覧を確認できる．

ここで記述した樹形図状に広がるファイルのトレースには追跡情報のログにコピー（派生）元とコピー（派生）先の情報があれば，それを処理することで実現できる．簡単な実現方式として，操作元ファイルと操作先ファイルのファイル名を繰り返し検索する方式などがある．更に，この処理をクラウドのような大規模な範囲で行うための追跡の並列化手法とその時のアーキテクチャは[1],[2]で提案されている．

3 課題解決への検討

本章では2章で紹介した既存技術の課題に対するアプローチを整理する．2.2節のトレース処理については [1],[2]で解決しているものとし，2.1節のログ生成処理における，汎用性問題と精度問題の解決方法を検討する．

3.1 汎用性の高いログ生成

既存ログ生成技術の汎用性が低かった原因は，ログ生成の際ファイル操作環境に事前にエージェントのインストールが必要な事，2.1.2節では監視エージェントがアプリケーションに特化していたことにある．汎用性の向上法として，エージェントのインストールを必要としないログ生成処理を検討し，汎用アプリケーションに対する監視方式は監視精度に関連する項目として，3.2節で紹介する．

ログ生成はコンピュータ上で行われる処理であり，事前にインストールされたエージェントを用いない場合，外部から持ち込まれたプログラム（以降**ログ生成プログラム**と呼ぶ）を実行し，それによって追跡情報を生成させる必要がある．このため，本節での課題はいかにファイル展開と同時に**ログ生成プログラム**を持ち込ませるか，いかにファイル操作の契機に実行させるかの二点となる．**ログ生成プログラム**はファイルが操作される前までに持ち込まれている必要があり，ファイルとプログラムのセットをパッケージングして同時に持ち込ませる方式が有効である．実現可能な範囲でファイルと同時にプログラム

を持ち込ませ実行出来る構成として考案したものを以下に紹介する.

① ファイルをオープンするアプリケーション上で実行されるマクロ形式のログ生成プログラム
② ファイルオープン処理の脆弱性など，コンテンツ内のコード実行を伴う特定仕様を利用して実行されるログ生成プログラム
③ コンテンツを実行形式のファイルでラッピングし，ファイルオープン時に実行されるプログラム中に埋め込んだログ生成プログラム

これらのプログラム構成にはそれぞれ以下のデメリットがある.

① 対象となるアプリケーションはマクロを実行できるものに制限され，任意のアプリケーションで実現できないため汎用性がない．さらにセキュリティリスクを理由にファイルオープン時のマクロ自動実行は多くのアプリケーションで利用者に対してプログラムの実行同意を求めるようになっており，出所不明なマクロは実行されない可能性が大きい．情報漏洩の用途を考えると，漏洩ファイルを開く人が探知を警戒していることも想定されるので，ユーザの同意やアプリケーションの設定によってトレースプログラムの処理をキャンセルできる場合，目的とするプログラム実行契機とはならない.
② 脆弱性を用いるので，ウイルスとして削除される可能性があると同時に，脆弱性を塞がれるとトレースができなくなる可能性が高く，汎用的な方式とは言えない.
③ 目的に対する致命的なデメリットはない．ただし，ファイル形式が実行形式に変更されるため，実行時に警告が発せられるなど，トレースのための処理が実行されることを利用者に気づかずにトレースすることは難しい.

以上の契機とその効果分析から，目的の情報漏洩の追跡を対象に考えると，デメリットの少なさから③が有効と判断し，この方式による実現方法を4章で検討する.

3.2 高精度な操作監視

汎用的なアプリケーションの動作を監視する方式として，アプリケーション単体を対象に監視する動的プログラム解析と呼ばれる手法と，仮想マシンモニタを用いて利用者環境の OS とその上で実行されるすべてのアプリケーションのディスク，メモリアクセスを監視する方式に大別される.

アプリケーション単体を対象にした動的プログラム解析では解析対象 OS 上の解析プログラムを用いる．解析プログラムでは外部のアプリケーションを実行でき，実行時にその外部アプリケーションの挙動をメモリレベルで監視，解析できる．ファイルの追跡に利用した場合，ファイル情報とそれが読み込まれ，処理され伝搬していったメモリアドレスを監視でき，非常に高い精度でファイル情報の伝搬を追跡できる．動的プログラム解析ツールとしては DynamoRIO[4], Pin[5], Valgrind[6]などが知られており，それぞれ対応する環境，性能や定義しやすい監視対象に特徴がある．この解析手法を用いることで解析プログラムを経由して実行されるアプリケーションから，直接なされるメール送信やアップロード，印刷，別名保存などの出力は，出力境界点まで追跡できる情報をログとして出力できる．しかしデメリットとして，動的プログラム解析では監視対象アプリケーションから監視対象外アプリケーションへの情報伝搬をアプリケーションインターフェイスまでしか追跡できない．解析プログラム経由で実行されたアプリケーションのメモリやディスクに対するアクセスは監視対象となるが，例としてコピー&ペーストによる監視対象外アプリケーションへの流出についてはクリップボードなどの直接的なコピー先を特定するまでにとどまり，クリップボードを経由して伝搬される先については検知できない．これを解消するために，既に実行されているプロセスに動的プログラム解析をアタッチする必要があり，そのタイミングや対象プロセスの特定方法は別途考慮する必要がある.

仮想マシンモニタからシステム全体を監視

する方式では監視対象環境が仮想マシン上にある前提のもと，仮想マシンのホスト側での仮想マシンモニタにプログラムを仕掛ける．仕掛けたプログラムは仮想マシンモニタの命令をフェッチし，仮想マシンでの CPU やメモリ，IO 動作を取得，解析する．そのため，単体アプリケーションにとどまらず，ゲスト OS 全体について，どの情報がどこへ流通しているかを監視することができる．しかし，この解析手法では仮想マシンを用いる必要がある他，仮想マシンを用いた場合でも事前にホストマシン上に監視プログラムを仕掛る必要があり，実用に向けた課題は多数存在する．

これらの解析手法をデータフローの追跡に利用した研究としてテイント解析が知られている．テイント解析とは対象情報に対してテイントと呼ばれるタグと伝搬追跡ルールを付与し，そのタグのメモリ上での伝搬から，情報漏洩の特定やマルウェアの感染解析[7]，[8]を行う技術である．テイント解析を用いた情報漏洩に対する研究として，OS 上の動的プログラム解析を用いた研究では TaintEraser[9]，RIFLE[10]が知られており，VM に依存するものとしては TaintBochs[11], Panorama[12]がある．

本研究の目的である漏洩ファイルの追跡に対しては上記の方式の内，3.1節の汎用性も考慮すると環境への変更が少ない動的プログラム解析を用いた手法が有効と判断し，以下で実現方式を検討する．

4 提案方式

本研究では3章での検討を基に漏洩ファイルに対して行われた操作をトレースし可視化するため，テイント解析機能を搭載した自己展開型ファイル（図 4）を提案する．

3.1節の検討に従い，追跡対象となるファイルを実行形式の自己展開型ファイルへのラッピングし流通させる．流通経路にいるユーザはこの実行ファイルを開くことにより内包される本来開きたいファイルを展開（復号）する．この時，展開（復号）したファイルは3.2節で記載した，OS 上で実行されるテイント解析プログラムを経由して対応するアプリケーションプログラムによりオープンされる．テイント解析プログラムではファイルをオープンする処理が行う操作を監視し，メモリレベルという高精度で伝搬を検知する．

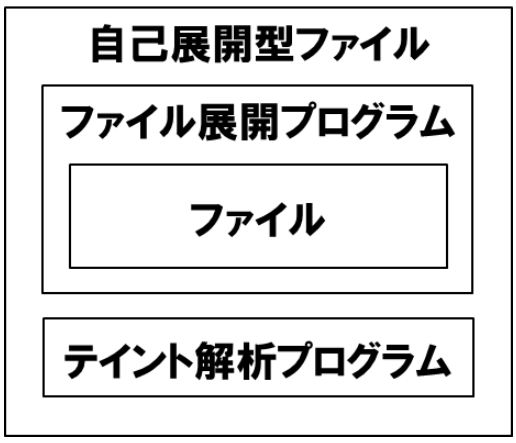

図 4 自己展開型ファイル

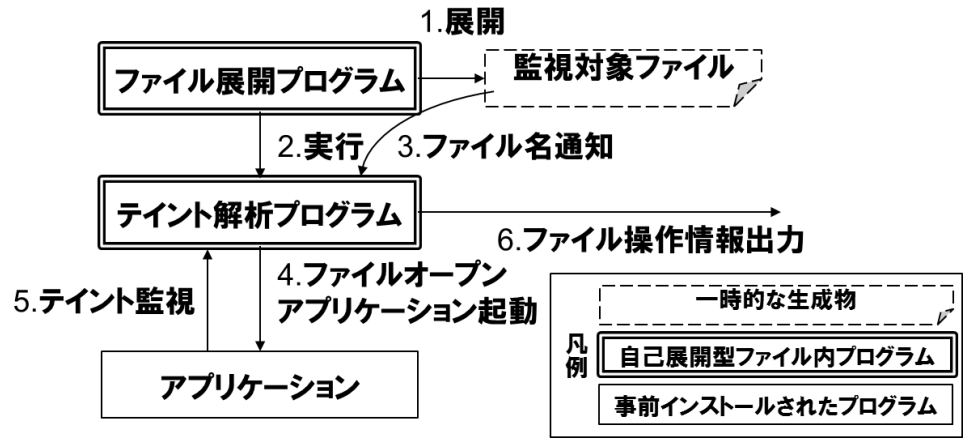

図 5自己展開型ファイルのオープン時挙動

ユーザが自己展開型ファイルをオープンした時の内部処理の流れは図 5 のとおりである．まず，ファイル展開プログラムが内包されている監視対象ファイルを展開する．

その後，展開し

たファイル名を渡しながらテイント解析プログラムを実行する. 実行されたテイント解析プログラムは受け取ったファイルをオープンすべきアプリケーションを特定し, そのアプリケーションを解析対象として起動する. その後, 解析対象のプログラムが終了するまで, 監視を続ける. 解析と同時に, 境界点を超えた外部への伝搬を検知した際は, ファイル操作の追跡情報として出力する.

この一連の処理によって, 自己展開型ファイルとして漏洩ファイルを実行できる環境下で, 目標としていたファイルに対する操作を記録, 追跡ができる.

5 提案方式の残課題

4章の提案方式によって, 本研究の目的である漏洩ファイルのトレースと可視化機能の実現は机上で確認できた. しかし, 実用化のために目的とする機能以外にも検討すべき課題があり, 残課題として, 本章で整理する.

・自己展開型ファイルのセキュリティ保証: 自己展開型ファイルは実行形式のファイルとなる. これを真似て自己展開型ファイルを偽り, マルウェアを実行させる攻撃が考えられる. 対策として電子署名等が考えられるが, 自己展開型ファイルに再変換するなどの処理をする場合, 再変換の際に秘密鍵を用いて再署名しないといけない. しかし, 署名のための秘密鍵をファイル自身に含めることはできないため, 構成上の工夫を要する.

・テイント解析プログラムのサイズ削減: テイント解析のプログラムは一般的に容量が大きく (例えば Pin の場合？？MB), 自己展開型ファイルに内包させることが困難である. 特にメールなどで流通できるファイルサイズに対して, テイント解析のプログラムを同梱するためには, サイズの圧縮が必要となる. 対策として自己展開型ファイル内にテイント解析のための全データを保管せず, 利用時にネットワークからダウンロードする方式が考えられる. ただし, その場合でもファイルオープンの際にダウンロードする情報量を削減することが望ましい.

・テイント解析の高速化: テイント解析技術自体まだ研究段階であり処理時間が 1.4[9]～300 倍とオーバヘッドが大きく, 用途によっては実用に向けてはさらなる高速化が必要となる.

・テイント解析を経由しない内包ファイルへのアクセス制限: 自己展開型ファイル内のコンテンツファイルに対して、追跡情報の生成回避を防ぐ必要がある. ファイル展開プログラムのリバースエンジニアリングや展開したファイルの横取りなど, 多様な攻撃が考えられる. 既存の IRM 技術を利用することで達成可能な部分もあるが適用方式と達成度を評価する必要がある.

・ファイル操作情報の出力方法: ファイルをオープンする際の環境がオンライン環境であれば, ネットワーク経由で追跡情報を送信することができるが, オフライン環境ではファイル操作情報の送信ができない可能性がある. 特に検知を警戒している攻撃者であれば, 検知を避けオフラインでファイルを閲覧する可能性が高い. オフラインでのファイル展開を許容すると, ログ蓄積, その後のログ送信に対する妨害や偽装, テイント解析プロセスへの攻撃など把握できないセキュリティリスクが大幅に増加する. しかし, 利便性を考慮すると, オフラインでの展開も必要な場合も有り, 安全性とのそのトレードオフとして展開, 実行に対する制限や対策のさらなる考察が必要である.

・単一アプリケーションの監視の境界点を超えた追跡: 3.2節で記載した, コピー&ペースト先アプリケーションの追跡など, テイント解析の監視可能範囲を拡大し, 追跡すべきアプリケーションの特定及びテイント解析下での実行方法の検討が必要である. また, アプリケーション終了後も別名保存された監視対象ファイルを追跡する場合, 例えば自己解凍型ファイルへコンバートする等の方法の検討も必要である。

これらのように, 空間的・時間的観点での追跡範囲を整理し, 境界点から外に渡る情報に対するトレースの可能性と実現方法を検討する必要がある.

6　まとめ

本研究ではファイルが漏洩した際の追跡方式を提案した．この方式を用いることで，内包するテイント解析プログラムを実行できる任意の環境でファイルの漏洩後の状態を追跡できる．この技術により，情報漏洩先の特定ができると，情報漏洩後の削除要求等が可能になり，一度漏洩ファイルが永久にインターネット上に出回る事態を回避できると考えられる．しかし，提案した方式の実現には解決すべき課題が多数残っており，今後も検討を重ねることで実現を目指したい．

参考文献

[1]　Iifan Tyou, Shinichi Nakahara: The Accountability of Cloud Services and Traceability Technology, APNOMS, 2012

[2]　張　一凡，竹内　格: MapReduce を用いたログ間の依存関係ツリーの抽出アルゴリズムの提案,ipsj 第 74 回全国大会

[3]　IPA：標的型サイバー攻撃の事例分析と対策レポート,
http://www.ipa.go.jp/files/000024536.pdf

[4]　DynamoRIO, http://dynamorio.org/

[5]　intel: Pin,
http://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool

[6]　Valgrind Developers: Valgrind,
http://www.valgrind.org/

[7]　Yuhei Kawakoya: VM-Based Malware Detection System, 16th USENIX Secuirty Symposium,
WORK-IN-PROGRESS(WIPS) ,Boston August 10,2007

[8]　Eitaro Shioji, Yuhei Kawakoya, Makoto Iwamura, Takeo Hariu: Code Shredding: Byte-Granular Randomization of Program Layout for Detecting Code-Reuse Attacks, 28th Annual Computer Security Applications Conference(ACSAC),Orlando, Florida, USA December 2012

[9]　David Zhu, Jaeyeon Jung, Dawn Song, Tadayoshi Kohno, David Wetherall: TaintEraser: protecting sensitive data leaks using application-level taint tracking,

[10] Neil Vachharajani, Matthew J. Bridges, Jonathan Chang, Ram Rangan, Guilherme Ottoni, Jason A. Blome, George A. Reis, Manish Vachharajani, David I. August: RIFLE: An architectural framework for user-centric information-flow security, MICRO, 2004.

[11] Jim Chow, Ben Pfaff, Tal Garfinkel, Kevin Christopher, Mendel Rosenblum: Understanding data lifetime via whole system simulation, USENIX Security Symposium, 2004

[12] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda: Panorama: capturing system-wide information flow for malware detection and analysis. In CCS, 2007.

[13] Syslog:
http://tools.ietf.org/html/rfc3195

[14] Flume: http://flume.apache.org/

[15] HTML5: http://www.w3.org/TR/html5/