攻撃回避のためのファイル不可視化手法の提案

佐藤 将也[1]　山内 利宏[1]　谷口 秀夫[1]

概要：セキュリティソフトウェアやプログラムの動作を記録するサービスは攻撃者にとって不都合であり無効化される可能性がある．これらの重要サービスを攻撃から保護することは，攻撃による被害拡大を防止するために重要である．重要サービスに関する情報にはプロセス情報とファイルがあり，これらを攻撃者から不可視化することで攻撃を回避し，重要サービスを攻撃から保護できると考えられる．本稿では，重要サービスに関するファイルの不可視化手法を提案する．具体的には，重要サービスからのみ参照可能なファイルを提供する機能を実現する．これにより，重要サービスに関するファイルをもとにした重要サービスの特定と攻撃を回避する．

キーワード：攻撃回避，ファイルアクセス制御，仮想化技術

1. はじめに

マルウェアによる攻撃への対策は重要な研究課題になっており，攻撃の被害を防ぐためのセキュリティソフトウェアが多く研究開発されている．また，攻撃の検知や攻撃による被害の確認のためにプログラムの動作を記録するソフトウェアが広く利用されている．本稿では，これらのソフトウェアを重要サービスとする．これらの重要サービスを攻撃から保護することは，攻撃による被害拡大を防止するために重要な研究課題である．攻撃者は，重要サービスを攻撃するために，重要サービスの存在を特定し，重要サービスを検出すると，サービスごとに攻撃を行う．重要サービスへの攻撃を防止する手法 [1], [2], [3] が多く研究されているものの，重要サービスの存在の特定を回避する手法は十分な対処がなされていない．

重要サービスの特定方法には，重要サービスを提供するプロセスの検知と重要サービスの関連ファイルをもとに特定する方法が考えられる．著者らはこれまでに，重要サービスを提供するプロセスの検知への対処として，重要サービスに関するプロセス情報の不可視化手法 [4], [5] を提案した．しかし，重要サービスの関連ファイルをもとにした特定の回避は未対処である．

本稿では，重要サービスの関連ファイルをもとにした重要サービスの特定を回避し攻撃を回避するために，ファイル不可視化手法を提案する．具体的には，重要サービスからのみ参照可能なファイルを提供する機能を実現する．重要サービスのサービス提供を妨害しないために，重要サービスからは従来通りに関連ファイルの参照を可能にする．一方，重要サービス以外からは，当該ファイルの参照を不可能にする．これにより，関連ファイルをもとにした重要サービスの特定を困難にする．本稿で提案するファイル不可視化手法には，仮想化技術を用いる．重要サービスを走行させる仮想計算機（Virtual Machine, 以降，VM）とは異なるファイル提供 VM を用意し，ファイル提供 VM 上に関連ファイルを配置する．重要サービスからのファイル操作要求を仮想計算機モニタ（VM Monitor, 以降，VMM）により監視し，ファイル提供 VM 上で動作する代理プロセスにファイル操作要求を転送する．代理プロセスは，受け取ったファイル操作要求を処理し，結果を VMM を介して重要サービスに返却する．以上の手順により，重要サービスからのみ参照可能なファイルを実現する．

2. 研究背景

2.1 重要サービスの保護

ANSS [1] では，Windows において API を監視するドライバを導入することで，セキュリティソフトウェアを停止させる API を検知するとセキュリティソフトウェアを終了させることなく API を終了させる．これにより，マルウェアによるセキュリティソフトウェアの停止を防止している．

仮想化技術を用いたセキュリティソフトウェアの保護手法として，VM 外部にセキュリティソフトウェアを移行し，外部の VM あるいは VMM から VM 内部の監視を行

[1] 岡山大学 大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

う手法として VMI [2] が提案されている．この手法では，VM は VMM とは隔離された環境で動作するという前提のもと，VM 内部のセキュリティソフトウェアを VM 外部で動作させることで，セキュリティソフトウェアへの攻撃を防止している．ただし，既存のセキュリティソフトウェアを VM 外部に移行しただけでは，VM 内から見える情報と VM 外から見える情報が異なる問題（セマンティックギャップ）が存在する．

また，重要サービスの保護を目的としていないものにも，仮想化技術を用いて機密ファイルを保護する手法として Filesafe [6] がある．Filesafe は，VMM 内にポリシを持ち，ポリシにしたがって VMM レベルでアクセス制御を行う．これにより，VM 上の OS が攻撃を受けた場合でもポリシを強制できるため，機密情報の漏えいを防止できる．

2.2 既存研究の問題点

ANSS のように VM 内部にセキュリティ機構を導入した場合，カーネルレベルでの攻撃により無効化される可能性がある．VMI と同様にセキュリティ機構を VM 外部に移植することで VM 上の OS が攻撃を受けてもセキュリティ機構自体の安全性に影響はないものの，既存セキュリティソフトウェアの再利用が困難である．また，セキュリティソフトウェアが特定の VM 上に集中するため，その VM に負荷が集中する問題やその VM が単一障害点になる問題が考えられる．Filesafe はファイルの読み込みと書き込みをポリシにより制御可能としているものの，ファイルの存在自体は可視であるため，関連ファイルをもとに重要サービスを特定される可能性がある．

以上より，VM 外部にセキュリティソフトウェアを配置する手法がソフトウェアの隔離という点で有効である一方で，既存ソフトウェアの再利用が困難であるという問題がある．

また，著者らの調査した限りでは，重要サービスを不可視化することで攻撃を回避する研究はないものの，攻撃を受ける前段階で重要サービスの存在を攻撃者から不可視にすることで攻撃を回避するという観点は重要であると考える．VMI は重要サービスを VM 外部に移植する点で VM 内部からは重要サービスを不可視化できているものの，上述の問題点がある．

3. ファイル不可視化手法

3.1 目的

(目的) 重要サービスの関連ファイルをもとにした重要サービスの特定の回避

重要サービスを特定する方法として，走行中プロセスのプロセス情報を参照する方法への対処をこれまでに提案した．しかし，プログラムによっては，設定ファイルが存在し，設定ファイルの存在から重要サービスを特定される可能性がある．例えば，デーモンの中には，走行中は PID を指定するファイルを作成するものがある．また，設定ファイルを改ざんされると，重要サービスの動作を変更される恐れがある．例えば，セキュリティソフトウェアの中には，ファイルをスキャンしてマルウェアの利用するファイルと一致するものを検知するものがある．この際，ファイルをスキャンするためのポリシのホワイトリストをマルウェアに操作されると，マルウェアが存在しているにもかかわらず検知できなくなる問題がある．

そこで，重要サービスの関連ファイルをもとにした重要サービスの特定を回避する方法を提案する．具体的には重要サービスからのみ参照可能なファイルを作成可能にすることで，重要サービスの動作を妨げることなく，他のプロセスからは関連ファイルを不可視化し，関連ファイルをもとにした重要サービスの特定を回避する機構を実現する．

3.2 要件

(要件 1) 重要サービスの関連ファイルを攻撃者から不可視にすること

(要件 2) 重要サービスの関連ファイルへの不可視化を他のプロセスやカーネルモジュールから検知されないこと

ファイルをもとにした重要サービスの特定を回避するために，重要サービスを攻撃者から不可視化する必要がある．これにより，関連ファイルをもとにした重要サービスの特定を困難にする．また，本研究の目的は，重要サービスの関連ファイルをもとにした重要サービス特定の回避である．このため，重要サービスの関連ファイルの存在を攻撃者から検知されないために，重要サービスの関連ファイルへの不可視化を攻撃者から検知されてはならない．

3.3 ファイルの不可視化

3.3.1 課題

要件を満たすための課題として以下がある．

(課題 1) 重要サービスの関連ファイルに対するファイルアクセスを捕捉できること

(課題 2) ファイルアクセスを制御できること

(課題 3) (課題 1) と (課題 2) への対処を重要プロセス以外のプロセスやカ―ネルモジュールから検知されないこと

(課題 1) と (課題 2) は，(要件 1) を満たすために必要である．重要サービスの関連ファイルを攻撃者から不可視化するには，重要サービスの関連ファイルへのファイルアクセスを捕捉し，そのファイルアクセスを制御できる必要がある．また，(要件 2) を満たすために，(課題 1) と (課題 2) への対処が重要プロセス以外のプロセスや攻撃に利用される可能性のあるカーネルモジュールから検知されない必要がある．

3.3.2 重要サービスの関連ファイルに対するファイルアクセスの捕捉

ファイルアクセスの捕捉手法として以下が考えられる．

(1) ライブラリ呼び出しの捕捉
(2) システムコールの捕捉
(3) ファイルシステムの処理の捕捉
(4) デバイスアクセスの捕捉

通常のファイルアクセスは，ライブラリ関数が呼び出され，ライブラリを経由してカーネルに処理を依頼するシステムコールが発行される．その後，カーネル内でファイルシステムを経由してデバイスドライバが呼び出され，ディスクへアクセスが発生する．

本稿では，システムコールを捕捉することでファイルアクセスを監視する．本稿における目的である重要サービスの関連ファイルの不可視化のためには，ライブラリ呼び出しでファイルアクセスを捕捉し制御することが望ましい．しかし，ライブラリは動的にリンクされている場合と静的にリンクされている場合がある．動的にリンクされている場合は，メモリ上の特定の関数呼び出しを監視すれば，ファイルアクセスを捕捉できる．一方で，静的にリンクされている場合は，プログラムごとに監視すべき関数呼び出しのメモリ上の場所が異なる．このため，すべてのライブラリ呼び出しを捕捉するのは困難である．

一方で，ファイルシステムやデバイスドライバによるファイルアクセスを監視すれば，すべてのファイルアクセスを監視できる．しかし，これらの処理は，ファイルアクセス経路の最終段階である．このため，これらの処理に到達する前にファイルアクセスの要求内容を解析されると，重要サービスの関連ファイルへのアクセスが攻撃者から検知されてしまう．

以上より，ファイルアクセスにおいて利用されるシステムコールを捕捉することで，(課題 1) へ対処する．

また，(課題 3) への対処として，攻撃者がファイルアクセスを監視するためにライブラリ関数の呼び出しを監視する場合を考慮し，ライブラリが配置されているメモリ領域の書き込みの禁止とライブラリのファイルの完全性検証を行う．これにより，ブレークポイントの挿入やライブラリの書き換えによるファイルアクセスの監視を防止する．さらに，デバッグレジスタの利用によるライブラリ関数呼び出しの監視については，デバッグレジスタの利用を制限することで対処する．

3.3.3 ファイルアクセスの制御

3.3.2 項で述べたとおり，提案手法では，システムコールを捕捉し，ファイルの不可視化を実現する．このため，本項では，システムコールを捕捉した際におけるファイルアクセスの制御手法を述べる．

ファイルアクセスは，通常，ファイルハンドラを取得し，そのファイルハンドラに対して操作を行う．このため，ファイルハンドラの取得の可否により，ファイルアクセスを制御する．提案手法では，ファイルハンドラを取得する処理を捕捉し，対象となるファイルが重要サービスの関連ファイルである場合には，システムコール発行元のプロセスを確認する．システムコール発行元のプロセスが重要サービスの場合は，ファイルハンドラを返し，発行元のプロセスが重要サービスでない場合は，ファイルハンドラの取得を失敗させる．

また，ファイルハンドル取得後のファイルアクセスを捕捉し，制御することで，重要サービスの関連ファイルを重要サービス以外から参照不可能にする．詳細は 3.3.4 項で述べる．

3.3.4 外部の VM へのファイルの配置とファイルアクセス

提案手法では，(課題 3) を満たすために，仮想化技術を用いる．具体的には，監視対象の OS を監視対象 VM 上で動作させ，重要サービスの関連ファイルの実体を異なる VM（ファイル提供 VM）に配置する．これにより，root 権限を持つプロセスやカーネルモジュールを攻撃者に乗っ取られた場合においても，重要サービスの関連ファイルをもとにした重要サービスの特定を困難にする．また，ファイルの実体を VM 内に配置しないことで，ディスク全体を探索したとしても，重要サービスの関連ファイルは存在しないため，重要サービスの検知は困難である．ただし，ファイル提供 VM に重要サービスの関連ファイルを配置すると，これらの関連ファイルへ重要サービス自身がアクセスできない問題がある．

そこで，重要サービスからのファイルアクセスをファイル提供 VM に転送する手法を用いる．具体的には，ファイル提供 VM 上に代理プロセスを配置する．提案手法を適用した場合のファイル配置と代理プロセスへのシステムコール情報の転送の様子を図 **1** に示す．本稿では，重要サービス以外のプログラムを通常サービスとする．重要サービスからのファイルアクセスは，VMM により捕捉され，代理プロセスに転送される．代理プロセスは，受け取った要求を重要サービスの代理として実行し，重要サービスの関連ファイルにアクセスする．代理プロセスはファイルアクセスの結果を VMM 経由で重要サービスに返却する．

以上より，重要サービスから関連ファイルへのアクセスを可能にし，かつ重要サービス以外からは不可視のファイルを提供する．これにより，(課題 3) に対処する．

4. 実現方式

4.1 想定する環境

本稿では，Intel VT-x を用いて仮想化された環境で監視対象 VM が走行することを想定する．また，監視対象 VM 上では，sysenter によりシステムコールが実行される環境を想定する．

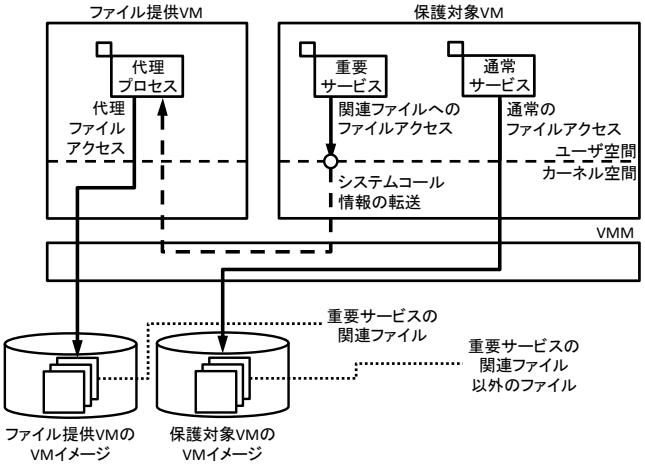

図 1　提案手法を適用した場合のファイル配置とシステムコール情報の転送

4.2　システムコールの監視

sysenter は，x86 アーキテクチャで提供される高速システムコール命令であり，ユーザモードで発行されると，モード切替を行い，`sysener_eip_msr` レジスタに格納されたアドレスへ命令ポインタをセットする．この際，`sysenter_eip_msr` のアドレスを書き換えておくことで，ページ例外を発生させる [7]．VT-x により仮想化された環境では，ページ例外が発生すると，VM exit により処理が VMM へ遷移する．この際，VMM が特定のアドレスにおけるページ例外を検知することで，システムコールの発行を監視できる．システムコールの引数や動作中プロセスの情報は，レジスタ値をもとにゲスト OS のメモリを解析することで取得できる．

4.3　ファイルアクセスに関するシステムコールの制御

4.3.1　ファイルハンドラの取得

3.3.3 項で述べた通り，ファイルハンドラの取得を要求するシステムコールの結果を変更することで，重要サービスの関連ファイルを重要サービス以外から不可視化する．提案手法では，ファイルハンドラの取得を行うシステムコールの発行を監視し，発行元が重要サービス以外の場合は，即座にゲスト OS に処理を返却し，通常のシステムコール処理を再開させる．ファイルの実体は，ファイル提供 VM に存在するため，通常のシステムコール処理を行ったとしても，重要サービスの関連ファイルは，重要サービス以外からは検知されない．

4.3.2　ファイルの読み込みと書き込みの制御

提案手法では，重要サービスが発行するシステムコールのうち，ファイルの読み込みと書き込みを制御する．4.4 節で後述するように，提案手法では，重要サービスの関連ファイルは既知であるとする．このため，新たにファイルを作成することや既存の重要サービスの関連ファイルを削除することは想定しない．

重要サービス以外からは，重要サービスの関連ファイルのファイルハンドラを取得できないため，ここでは，重要サービスが関連ファイルへの読み込みと書き込みを行う場合の処理を示す．重要サービスがファイルの読み込みや書き込みを行うシステムコールを発行した場合，VMM がシステムコールを検知し，代理プロセスにシステムコール名と引数を転送する．代理プロセスは，システムコール名と引数をもとに，重要サービスの関連ファイルにアクセスし，結果をバッファに格納する．VMM は，代理プロセスが結果を格納した領域を重要サービスの用意したバッファに返却する．この際，代理プロセスの処理の成否に応じて処理を変更する．具体的には，システムコール発行時の後処理を VMM でエミュレートし，成否に応じた結果をレジスタやメモリに書き込み，システムコール処理の最後へ命令ポインタを設定し，VM に処理を返却する．これにより，監視対象 VM 上のカーネル処理をバイパスし，重要サービスの関連ファイルへのアクセスを実現する．

4.4　重要サービスの関連ファイルの設定

提案手法では，重要サービスの関連ファイルは既知であることを想定する．提案手法では，重要サービスの関連ファイルとして，主に設定ファイルを想定している．多くの場合，設定ファイルは，特定のディレクトリ以下に配置されることが多く，また，利用されるファイルは重要サービスのインストール時に決定される．このため，重要サービスのインストール時に作成されるファイルをファイル提供 VM に移行し，VMM はファイルアクセスを監視する際に，ファイル提供 VM に移行したファイルの名前を監視対象とする．

5.　関連研究

VMI と同様にセキュリティソフトウェアを VM 外部に移行する手法として，Process Out-Grafting [3] がある．Process Out-Grafting では，セキュリティソフトウェアを監視対象の VM とは異なる VM 上で動作させ，監視対象のソフトウェアをセキュリティソフトウェアと同じ VM 上で動作させることで，セキュリティソフトウェアを隔離しつつ VM 外からの監視におけるセマンティックギャップの問題を解決している．また，Process Out-Grafting は既存のツールを改変する必要はない．一方で，複数 VM が走行する環境においては，セキュリティソフトウェアが特定の VM に集中するため，その VM に負荷が集中したり単一障害点になる問題がある．本稿における提案手法では，代理プロセスを配置する VM を分散させることで，単一の VM に負荷が集中することを回避できる．また，セキュリティソフトウェアの処理の多くは監視対象 VM 上で行い，ファイルアクセスのみ代理プロセスに転送するため，ファイルの読み書きを多く要求する処理を行わない限り負荷は監視

対象 VM 外には影響しない．

Filesafe [6] は，VMM を用いてファイルへのアクセス制御を行う．ただし制御するのは読み込みと書き込みのみであるため，ファイル存在自体は攻撃者から検知できる．提案手法は，ファイルの存在自体を攻撃者から不可視化するため，目的が異なる．

ファイルを不可視化する手法は，マルウェアによって利用されることが多い．特に，マルウェアの存在を隠す機能を持つルートキットで用いられる．ルートキットによるファイルの隠ぺいを検出する手法として GhostBuster [8] が提案されている．文献 [8] では，ファイル不可視化の検知手法として，API フックの検出とファイル一覧の取得手法ごとの結果の比較による検知の 2 とおりの手法を述べており，GhostBuster は後者である．GhostBuster は，オペレーティングシステムがルートキットに感染した環境を想定し，感染したオペレーティングシステム上で取得できるファイルの一覧と CD から起動したオペレーティングシステムで取得できるファイルの一覧を比較することで，ファイルの不可視化を検出している．本稿における提案手法は，ファイルの実体は異なる VM 上に存在するため，VM 上のファイル一覧の取得方法によって結果が異なることはない．ただし，重要サービスから取得したファイルの一覧が攻撃者により参照された場合には，GhostBuster と同様の検知手法により重要サービスの関連ファイルをもとに，重要サービスを特定される可能性がある．

6. おわりに

本稿では，攻撃回避のためのファイル不可視化手法の設計を述べた．提案手法では，VM 上で重要サービスを提供する環境を想定し，異なる VM に重要サービスの関連ファイルを配置する．また，VMM により VM 上のシステムコールの発行を監視し制御することで，重要サービスからのみ関連ファイルへのアクセスを可能にする．これにより，重要サービス以外のプロセスや VM 上のカーネルからは重要サービスの関連ファイルを不可視化し，重要サービスの関連ファイルをもとにした重要サービスの特定を困難にする．

残された課題として，重要サービスを提供するプログラムの実行ファイルの不可視化，および提案手法の実現と評価がある．

謝辞　本研究の一部は JSPS 科研費 16K16067, 16H02829 の助成を受けたものです．

参考文献

[1] Hsu F.H., Wu, M.H., Tso, C.K., Hsu, C.H. and Chen, C.W.: Antivirus Software Shield Against Antivirus Terminators, IEEE Transactions on Information Forensics and Security, Vol.7, No.5, pp.1439–1447 (2012).

[2] Garfinkel, T. and Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, Network and Distributed Systems Security Symposium, Vol. 3, pp.191–206 (2003).

[3] Srinivasan, D., Wang, Z., Jiang, X. and Xu, D.: Process Out-grafting: An Efficient "out-of-VM" Approach for Fine-grained Process Execution Monitoring Proc. 18th ACM Conference on Computer and Communications Security, pp.363–374 (2011).

[4] 佐藤将也，山内利宏，谷口秀夫：プロセス情報不可視化のための仮想計算機モニタによるメモリアクセス制御，情報処理学会シンポジウムシリーズ コンピュータセキュリティシンポジウム 2015 (CSS2015) 論文集, Vol.2015, No.3, pp.855–860 (2015).

[5] Sato, M., Yamauchi, T. and Taniguchi, H.: Process Hiding by Virtual Machine Monitor for Attack Avoidance, Journal of Information Processing, Vol.23, No.5, pp.673–682 (2015).

[6] Junqing W., Miao Y., Bingyu L., Zhengwei Q. and Haibing G.: Hypervisor-based Protection of Sensitive Files in a Compromised System, Proc. 27th Annual ACM Symposium on Applied Computing, pp.1765-1770 (2012).

[7] Dinaburg, A., Royal, P., Sharif, M. and Lee, W.: Ether: Malware Analysis via Hardware Virtualization Extensions, Proc. 15th ACM Conference on Computer and Communications Security, pp.51–62 (2008).

[8] Wang, Y.M., Beck, D., Vo, B., Roussev, R. and Verbowski, C.: Detecting Stealth Software with Strider GhostBuster, Proc. 2005 International Conference on Dependable Systems and Networks (DSN'05), pp.368–377 (2005).