Computer Security Symposium 2017
23 - 25 October 2017

仮想計算機を用いた重要ファイル保護手法

佐藤 将也[1]　山内 利宏[1]　谷口 秀夫[1]

概要：攻撃防止や被害状況把握のために，セキュリティソフトウェアや証拠保全のためのプログラムなどの重要サービスを攻撃から保護することは重要である．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，著者らはこれまでに，重要サービスのプロセス情報の参照を困難にすることで検知を困難にし，攻撃を回避する手法を実現した．また，攻撃回避のための重要ファイル保護手法を提案した．この手法は，重要サービスからのみ参照可能なファイルを提供することで，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．本稿では，仮想計算機を用いた重要ファイル保護手法の実現方式を述べる．

キーワード：ファイル保護，攻撃回避，仮想化技術

1. はじめに

マルウェアがセキュリティソフトウェアからの検知を回避する手法や，セキュリティソフトウェアを無効化する攻撃が問題となっている [1], [2], [3], [4]．文献 [3] では，マルウェアが攻撃に用いる手段を Evasion, Disruption, Modification, および Stealing の 4 つのクラスに分類されている．それでの攻撃手段により検知や防止の難しさや攻撃による被害が異なるが，本研究では，特に，セキュリティソフトウェアが無効化された場合，攻撃の検知や防止が難しくなると考え，セキュリティソフトウェアへの攻撃に着目する．Evasion の例として，セキュリティソフトウェアやファイアウォールを停止する攻撃が報告されている．また，Modification では，既存のファイルの上書きや新しいファイルを作成する方法や設定ファイルを書き換えることで，攻撃を行う方法が報告されている．

一方で，コンピュータセキュリティにごまかしを用いる方法がある [5], [6]．文献 [5] では，ごまかしをコンピュータセキュリティに用いる方法が述べられており，システムの機能や状態をごまかすことで，攻撃を困難にしたり攻撃の監視を行う手法が述べられている．文献 [6] では，攻撃者の行動を解析するために，脆弱性を狙った攻撃を検知すると，攻撃が成功したように見せかける偽の情報を返すデコイの Web サービスに誘導する方式が提案されている．

著者らは，セキュリティソフトウェアなどの重要サービスへの攻撃の回避にごまかしを用いる方法を研究している．重要サービスへの攻撃は，攻撃の初期段階において，攻撃対象の計算機で動作するソフトウェアの種類やバージョンを特定することで行われる．重要サービスの検知に用いられる情報には，プロセスに関する情報とファイルに関する情報がある．著者らは，重要サービスを提供するプロセス（以降，重要プロセス）に関する情報を攻撃者から不可視化することで，攻撃を回避する手法を提案した [7], [8]．また，重要サービスの関連ファイル（以降，重要ファイル）を保護することで重要サービスの特定を困難にする方式を提案した [9]．しかし，その詳細な実現方式は述べられていない．

本稿では，重要ファイル保護手法について，VM として Linux を用いた場合の実現方式を述べる．提案方式は，重要ファイルを重要プロセスからのみ参照可能にし，他のプロセス（通常プロセス）からは参照不可にする．具体的には，仮想計算機モニタ（以降，VMM）を用い，仮想計算機（以降，VM）上のファイルアクセスを監視することで，重要プロセスからのファイルアクセスと通常プロセスからのファイルアクセスを区別する．さらに，重要プロセスから重要ファイルへのアクセスの際は，ファイルアクセスを他 VM に転送し，代理実行する．これにより，重要ファイルは VM レベルで隔離され，通常プロセスからは参照できず，また，改変も困難である．また，重要ファイル以外のファイル（以降，通常ファイル）については，通常どおりの処理を行わせることで，重要プロセスの処理を妨げない．これにより，重要ファイルを検知されず，また，改変の困難な機構を実現する．

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

© 2017 Information Processing Society of Japan

2. 攻撃回避のためのファイル保護手法

2.1 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と重要ファイルを参照する方法が考えられる．本稿では，重要ファイルの保護により，重要サービスへの攻撃を回避する手法を述べる．ここで，重要ファイルとは，プログラムの設定ファイルや走行中のプロセスを識別可能なファイルを想定する．たとえば，プログラムの設定ファイルが通常プロセスから参照可能な場合，攻撃者により通常プロセスを通じて設定ファイルをもとに重要プロセスを識別される可能性がある．さらに，設定ファイルを改変された場合，攻撃者の都合の良いように重要プロセスの動作を変更される可能性がある．以上のことより，重要ファイルを攻撃者から不可視化し保護することで，重要プロセスの特定を回避する方式を提案する．

2.2 ファイル保護手法

ファイル保護手法の全体像を図 **1** に示す．提案方式では，保護対象 VM 上のプロセスによるファイルアクアセスを監視し，重要プロセスから重要ファイルへのファイルアクセスの際には，ファイル提供 VM にファイルアクセスの代理実行を依頼し，ファイル提供 VM 上の重要ファイルを読み書きする．通常プロセスから重要ファイルへのファイルアクセスの際は，システムコールを失敗させる．これにより，重要プロセスからのみ重要ファイルにアクセスできる環境を実現することで，通常プロセスからの重要ファイルの参照により重要プロセスを検知されることを防ぐ．重要ファイルは仮想ディスク単位で分離されているため，保護対象 VM の仮想ディスクの内容が攻撃者によりすべて検査されたとしても，重要ファイルを検知できない．また，仮想ディスク単位で隔離しているため，重要ファイルが改変されることもない．

2.3 別 VM へのファイルの配置とファイルアクセス

提案方式では，重要ファイルを保護対象 VM とは異なるファイル提供 VM に配置する．このように，VM 単位でファイルの配置箇所を分割することで，攻撃者が保護対象 VM 上でディスクを走査した場合においても，重要ファイルは攻撃者から検知できない．さらに，ファイル提供 VM からしか重要ファイルへアクセスできないことから，保護対象 VM とファイル提供 VM が隔離されており，かつ代理プロセスが攻撃を受けない限りは，重要ファイルが不用意に改変されることはない．

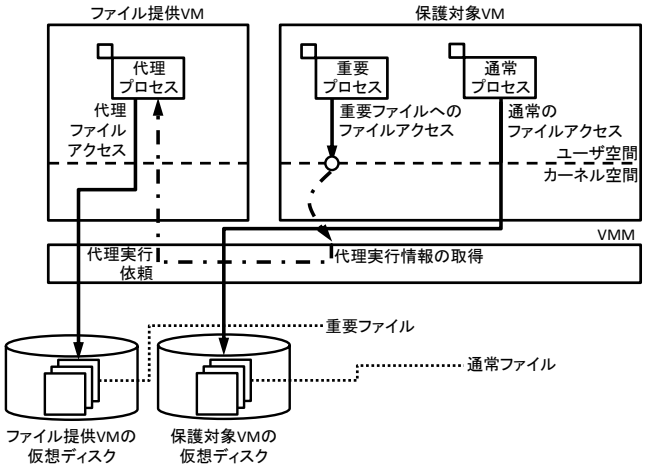


図 **1** 仮想化を用いたファイル保護手法の全体像

3. 実現方式

3.1 想定する環境

VMM として Xen を用い，保護対象 VM とファイル提供 VM では Linux を用いることを想定する．また，保護対象 VM は Intel VT-x を用いて完全仮想化されていることを想定する．

3.2 ファイルアクセス制御

3.2.1 ファイルアクセス制御の処理流れ

ファイルアクセス制御は，以下の 4 段階に分類できる．

(1) ファイルアクセスに関するシステムコールの捕捉
保護対象 VM 上のプロセスによるシステムコール発行を VMM により検知する．この際，システムコールを発行したプロセスが重要プロセスの場合は，システムコールの種類を識別する．システムコールの種類がファイルアクセスに関するものであった場合は処理を制御し，そうでない場合は通常の処理を継続させる．

(2) システムコールの判別とアクセス可否の制御
ファイルアクセスに関するシステムコールのうち，アクセス先のファイルが重要ファイルである場合はシステムコールの代理実行依頼を行い，そうでない場合は，通常の処理を継続させる．

(3) システムコールの代理実行依頼
システムコールの代理実行依頼に必要な情報を収集し，ファイル提供 VM に代理実行を依頼する．

(4) システムコール処理の返却
代理実行の結果を重要プロセスの利用するバッファに格納し，処理を返却する．また，アクセスの種別が通常ファイルなら，システムコール処理を継続させるために，保護対象 VM に処理を返却する．

重要プロセスの不可視化のためには，プロセス情報，ファイル情報，およびネットワークに関する情報を他プロセス

© 2017 Information Processing Society of Japan

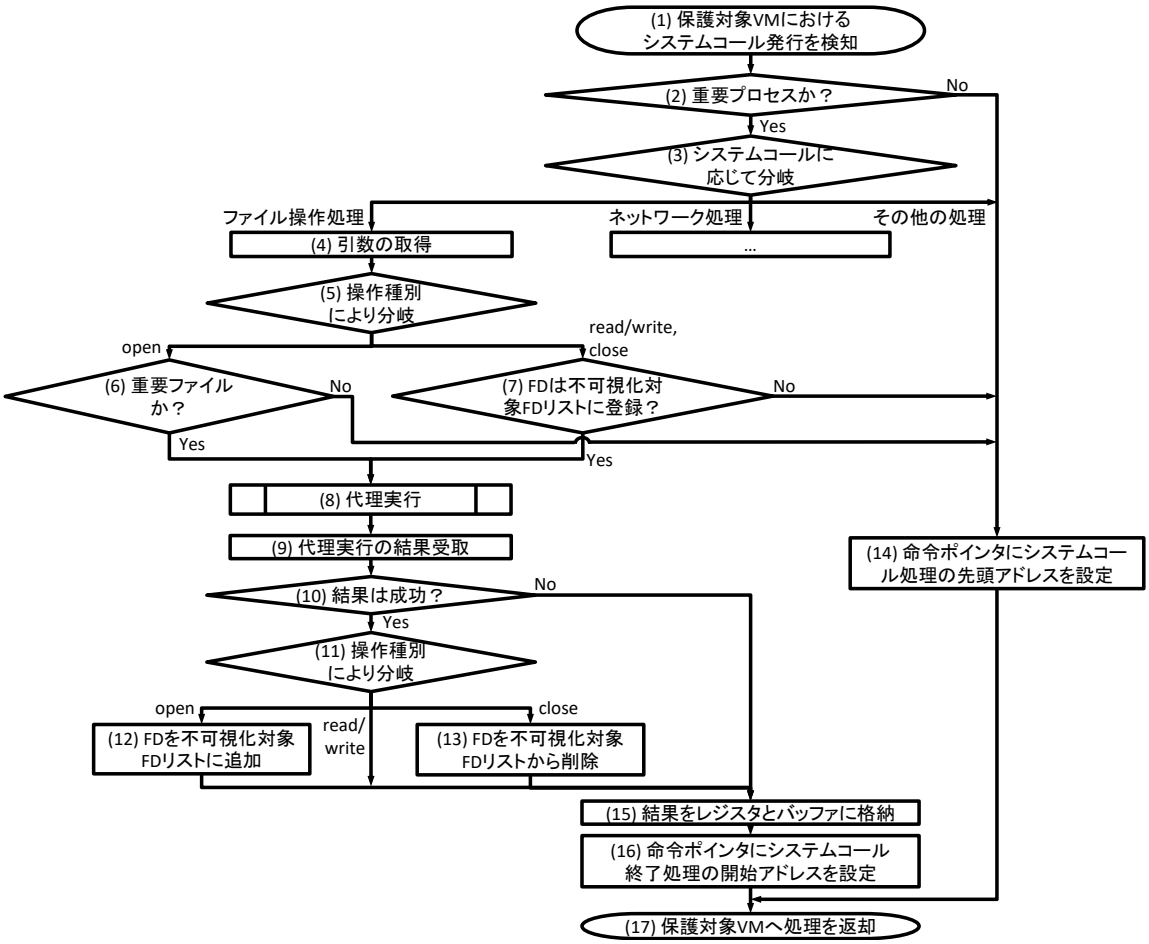


図 2 ファイルアクセス制御の処理流れ

等から不可視化する必要がある．本稿では，ファイル情報の不可視化についてのみ述べる．ネットワークに関する情報の不可視化は今後の課題とする．

3.2.2 システムコールの捕捉

保護対象 VM におけるシステムコールを VMM から捕捉するために，システムコール処理の先頭アドレス（SYSCALL 命令を用いてシステムコールを発行する場合は `LSTAR MSR` のアドレス）にハードウェアブレークポイントを設定することで，システムコールの発行を検知する．VMM は，デバッグレジスタを操作し，保護対象 VM 上で当該アドレスの命令が実行された場合に，デバッグ例外を発生させる．さらに，デバッグ例外の発生により VM exit を発生させることで，システムコールが発行される度に，VMM により検知し，システムコールの発行を捕捉する．

3.2.3 システムコールの制御

システムコールの制御について，処理の流れを**図 2** に示し，以下で述べる．

(1) 保護対象 VM におけるシステムコール発行により VM exit が発生し，VMM に処理が遷移する．VMM に処理が遷移した際に，VM exit の発生原因がページ例外であり，かつ例外の発生したアドレスが事前に設定したアドレスと一致した場合，提案方式の導入により発生したページ例外として，システムコール発行を検知する．

(2) システムコールを発行したプロセスが重要プロセスの場合は，システムコールの種類を判別する．システムコールを発行したプロセスが通常プロセスの場合は，(14) に移る．

(3) システムコール番号をもとにシステムコールの種類を判別し，種類に応じて分岐する．提案方式では，ファイルアクセスに着目しているため，ファイル操作処理の場合のみ，その後の制御に移行する．ネットワーク処理およびその他の処理を行うシステムコールの場合は，(14) に移る．

(4) システムコールの制御のために，引数を取得する．

(5) ファイル操作処理に関するシステムコールのうち，open の場合は (6) に，read/write, または close の場合は (7) に移る．

(6) open の対象が重要ファイルの場合，(8) に移る．open の対象が通常ファイルの場合，(14) に移る．

(7) read/write または close の引数で与えられた FD の値が重要ファイル FD リストに登録されている場合は，(8) に移る．登録されていない場合は，(14) に移る．

(8) ファイル提供 VM に代理実行を依頼する．代理実行の

© 2017 Information Processing Society of Japan

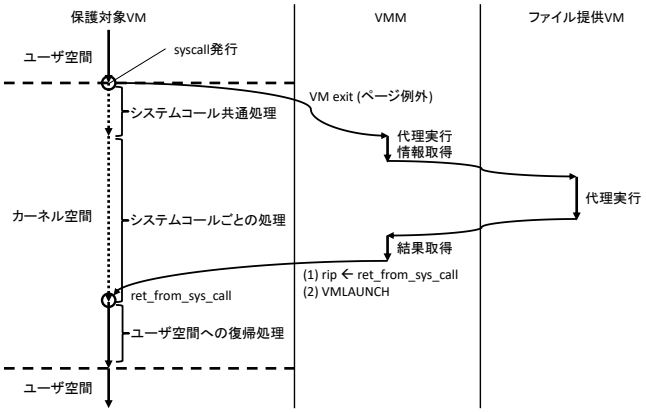


図 3 システムコール発行から復帰までのモード遷移

流れは 3.4.1 項で述べる．

(9) ファイル提供 VM から代理実行の結果を受け取る．

(10) 結果が成功の場合は，(11) に遷移する．結果が失敗の場合は，(15) に遷移する．

(11) 操作種別が open の場合は，(12) に移行する．操作種別が read/write の場合は，(15) に移行する．操作種別が close の場合は，(13) に移行する．

(12) 重要ファイルを open したため，重要ファイルの FD を重要ファイル FD リストに登録し，(14) に移る．

(13) 重要ファイルを close したため，重要ファイルの FD を重要ファイル FD リストから削除し，(14) に移る．

(14) システムコール処理を保護対象 VM 上で継続するために，保護対象 VM の命令ポインタにシステムコール処理の先頭アドレスを設定する．

(15) 代理実行したシステムコールの結果をレジスタに格納する．必要な場合は，保護対象 VM 上のバッファに結果を格納する．

(16) 命令ポインタにシステムコール終了処理の開始アドレスを設定する．この処理を実行した場合，保護対象 VM では，システムコール処理の開始直後にシステムコール終了処理が呼び出される．

(17) 保護対象 VM に処理を返却する．

3.2.4 システムコール処理の返却

保護対象 VM におけるシステムコール発行から保護対象 VM への復帰までのモード遷移を**図 3** に示す．重要プロセスによる重要ファイルへのアクセス後に，VMM から保護対象 VM に処理を返却する．この際，システムコールの開始処理から処理を再開させると，本来のファイルアクセス処理が行われる．重要ファイルを保護するためには，本来のファイルアクセス処理が行われないようにする必要がある．このため，提案方式では，重要プロセスが重要ファイルにアクスした際は，システムコール処理へ処理を返却する際に，システムコールの終了処理から開始させる．

3.3 重要ファイルの管理

保護対象 VM 上でプロセスがファイルへアクセスする際，提案方式では，アクセス元のプロセスとアクセス先のファイルの関係を把握する必要がある．さらに，保護対象 VM が複数存在する場合に，保護対象 VM ごとに，プロセスとファイルの関係を把握する必要がある．以上より，提案方式の実現には，以下の課題がある．

（課題 1）VM ごとのファイルの区別

（課題 2）プロセスごとのファイル操作の区別

（課題 3）重要ファイルと通常ファイルへの区別

（課題 1）については，VMID を用いることで対処する．VMID は，Xen において，VM を管理するために付与される識別子であり，VM 起動時に一意に割り当てられる．

（課題 2）については，CR3 レジスタに格納されたページディレクトリ（以降，PD）のアドレスを用いることで対処する．ここで，PID を用いず，CR3 レジスタの値を用いるのは，PID 取得のためのオーバヘッドを抑制するためである．PID を取得するためには，保護対象 VM 上のプロセス管理表を参照し，PID をメモリから取得する必要がある．一方で，CR3 レジスタの値は，VM exit 発生により VMM へ処理が遷移する際に退避されているため，PID のように保護対象 VM 上のメモリを参照する必要がなく，不要なオーバヘッドは発生しない．（課題 2）へ対処するには，プロセスを識別できればよく，PD のアドレスはプロセス固有であることから，（課題 2）へ対処できる．

（課題 3）については，未使用のディスクリプタ番号を大きい値から降順に利用することで対処する．重要プロセスは重要ファイルと通常ファイルの両方にアクセスすることが考えられる．このため，重要プロセスが利用するファイルディスクリプタのうち，重要ファイルと通常ファイルそれぞれに対応するディスクリプタ番号を管理する必要がある．ディスクリプタ番号はカーネルにより管理されているため，VMM から管理方法を変更し，重要ファイルと通常ファイルを区別させることは難しい．そこで，VMM が代理実行を行う場合は，ディスクリプタ番号を最大値から降順に割り当てる．これにより，重要ファイルと通常ファイルでディスクリプタ番号が衝突する可能性を提言でき，かつ，保護対象 VM 上のファイルディスクリプタの管理方法を改変する必要はない．（課題 3）に対処した際のファイルディスクリプタの管理方法を**図 4** に示す．

3.4 ファイルアクセスの代理実行

3.4.1 代理実行の処理流れ

図 5 にファイル提供 VM 上の代理プロセスにおける代理実行の処理流れを示す．

(1) 代理プロセスは VMM からの代理実行の処理依頼を受け付ける．

(2) 代理プロセスは，代理実行情報（3.4.2 項に後述）をも

© 2017 Information Processing Society of Japan

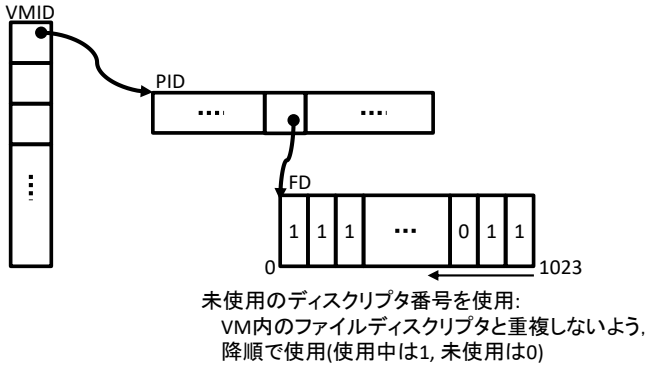


図 4 重要ファイルのファイルディスクリプタ番号の割り当て規則

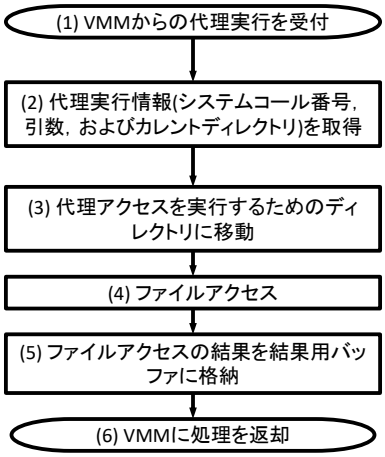


図 5 ファイルアクセス代理実行の処理流れ

とに，代理実行するシステムコールを決定し，引数を決定する．

(3) 代理プロセスは，代理実行情報のうち，PD と重要プロセスのカレントディレクトリの情報をもとに，ディレクトリを移動する．

(4) 代理プロセスは，(2) で作成した引数をもとにファイルアクセスを代理実行する．

(5) 代理プロセスは，(3) で実行した結果を結果用バッファに格納する．

(6) 代理プロセスは，VMM に処理を返却する．

以上の手順により，代理プロセスは重要プロセスのファイルアクセスを代理実行し，結果を VMM に返却する．

3.4.2 代理実行情報

VMM からファイル提供 VM 上の代理プロセスに代理実行を依頼する際に渡す情報を**表 1** に示す．

VMM 上では複数の VM が動作する環境を想定する．このため，VMID を用いて，代理実行の依頼元の VM を識別する必要がある．このために，VMID を用いる．VM 上で複数のプロセスが動作する環境を想定する．複数のプロセスのうち，重要プロセスごとにアクセス可能なファイルが異なり，さらに，重要プロセスと通常プロセスを識別する必要がある．このため，PD を用いる．重要プロセスのカレントディレクトリは，重要ファイルのパスを求めるために用いる．例えば，open の引数としてファイルのパスが相対パスで指定された場合，プロセスのカレントディレクトリを用いてファイルのフルパスを識別する．システムコール番号は，代理実行を依頼するシステムコールを識別するために用いる．また，代理実行を依頼するシステムコールの引数を代理プロセスに渡す．

表 1 の情報のうち，(1) と (4) は，レジスタ値から取得可能な情報である．(2), (3), および (5) は，レジスタ値をもとに保護対象 VM のメモリから取得する．

表 1 代理実行情報

通番	名前	説明
(1)	VMID	重要プロセスの動作する VM を識別するために用いる．
(2)	PD	重要プロセスを識別するために用いる．
(3)	重要プロセスのカレントディレクトリ	重要プロセスのアクセス先ファイルが相対パスで指定された場合に，アクセス先ファイルを識別するために用いる．
(4)	システムコール番号	重要プロセスが要求するファイル操作に関するシステムコールを識別するために用いる．
(5)	システムコールの引数	システムコールに渡す引数として用いる．

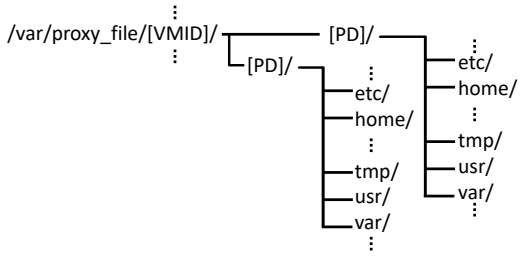


図 6 ファイル提供 VM のディレクトリ構成

3.5 ファイル提供 VM のディレクトリ構成

ファイル提供 VM において保護対象 VM のファイルを管理する際のディレクトリ構成を**図 6** に示す．ファイル提供 VM では，自 VM が利用するファイルのみアクセスできるように VM ごとにディレクトリを作成する．また，その中で重要プロセスが自身の利用するファイルのみアクセスするように重要プロセスごとにディレクトリを作成する．VM と重要プロセスは，`VMID` と `PD` で識別する．

3.6 重要ファイルの指定

VMM は，重要プロセスによるファイルアクセスを検知した際，アクセス先のファイルが重要ファイルか否かをもとに処理を制御する．このため，VMM は，重要ファイルの一覧を保持する必要がある．そこで，以下の方針で重要ファイルを決定する．

© 2017 Information Processing Society of Japan

表 2　重要プロセスと通常プロセスによる重要ファイルと通常ファイルへのアクセス可否の関係

	重要ファイル	通常ファイル
重要プロセス	読み書き可	読み込み可，書き込みにより重要ファイルに変更
通常プロセス	読み書き不可	読み書き可

(1) 事前に定義された重要ファイルの一覧
設定ファイルのように，事前にファイル名が静的に決定されているファイルは，事前にファイル名を指定できる．このため，事前に定義されたファイルの一覧をVMM が保持する．具体的には，/etc 以下の特定のディレクトリを指定し，そのディレクトリ以下のファイルをすべて重要ファイルとする．

(2) 新規に作成されたファイル
重要プロセスが新規に作成するファイルは，プログラムにより異なる．このため，事前に静的に決定するのは難しい．重要プロセスにより生成されたファイルは，重要プロセスの識別に利用可能な情報が含まれる可能性がある．このため，本方式では，重要プロセスが新規に作成するファイルはすべて重要ファイルとする．

(3) 上書きまたは追記するファイル
重要プロセスがファイルに上書きまたは追記を行った場合，書き込んだ内容から重要プロセスを特定される可能性がある．そこで，重要プロセスにより上書きまたは追記の対象となるファイルは，すべて重要ファイルとする．

重要プロセスと通常プロセスによる重要ファイルと通常ファイルへのアクセス可否の関係を**表 2** に示す．

3.7　提案方式による制限

提案方式では，ファイルディスクリプタを最大値から降順に利用する．このため，ファイルを同時に複数利用するようなプログラムでは，通常ファイルと重要ファイルのファイルディスクリプタ番号が衝突する場合がある．対処として，ファイルディスクリプタ数の最大値を大きく設定することで，通常ファイルと重要ファイルのファイルディスクリプタ番号が衝突する可能性を低くする方法がある．

4. 関連研究

セキュリティソフトウェアへの攻撃について対策手法が研究されている [1], [2]．文献 [2] では，セキュリティソフトウェアが攻撃される契機を分析し，攻撃されにくいセキュリティソフトウェアの構造について提案している．本手法は，セキュリティソフトウェア自体を改変する必要があるが，提案方式はセキュリティソフトウェアを改変する必要はない．ANSS [1] では，セキュリティソフトウェアを終了させる攻撃への対策として，プロセスを終了させるシステムコールの発行を監視し，終了の対象がセキュリティソフトウェアの場合は，システムコールを失敗させることで，セキュリティソフトウェアの終了を防止している．ANSS はカーネル空間での対策を提案しているが，提案方式は VMM による対策であり，カーネル空間と比較して攻撃が困難であるという特徴を持つ．

仮想計算機を用いたマルウェア解析手法が提案されている． [10], [11], [12], [13]．VMI [10] では，VMM から VM 内を観測可能な機構を実現することで，セキュリティソフトウェアを他 VM に移植する方法が提案されている．Ether [11] では，VMM を改変することで VM 上のシステムコール発行などを監視し，マルウェアの解析を行う環境を実現している．文献 [12] では，プロセスを他 VM に移植して動作させることで，解析環境を安全に保つ方法が提案されている．この方法は，解析対象のプロセスのユーザ空間でのコードは他の VM で動作させ，システムコールを元の VM に転送する．これにより，プロセス自身は元の VM で動作するようにしながら，ユーザ空間のコードを解析可能にしている．このため，元の VM が攻撃を受けて信頼できない場合でも，信頼できる解析環境を提供できる．Filesafe [13] は，ファイル保護のために VMM を用いる手法である．Filesafe は，VMM 内にポリシを保持し，ファイルの読み込みと書き出しを制御する．ファイルアクセスを制御するという点で提案方式と類似してるものの，提案方式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．提案方式は，これらの手法と同様に，仮想計算機を用いることで，セキュリティソフトウェアの保護を行う．

マルウェア解析のためにごまかしを用いる手法が研究されている [5], [6]．文献 [5] では，攻撃の解析や防御のためにごまかしを用いる手法が述べられている．具体的には，ハニーポットのように，攻撃を解析するために攻撃されるための環境を作成する手法や解析環境に誘導する方法が述べられている．文献 [6] では，脆弱性を狙った攻撃を検知した際に，攻撃が成功したように見せかける応答をすることで，先の段階の攻撃を解析する環境を実現している．具体的には，脆弱性を狙った HTTP 通信を検知した際に，解析用の環境を作成し，通信を転送する．解析用の環境は，脆弱性を狙った攻撃が成功したように見せかける偽の情報を応答することで，先の段階の攻撃を実行させ，攻撃の内容を詳細に解析する．提案方式は，ごまかしをセキュリティに用いる点で類似しているものの，攻撃の解析ではなくセキュリティソフトウェアの保護を目的としている点で異なる．

5. おわりに

仮想計算機を用いた重要ファイル保護手法について，Xen と Linux を対象とした実現方式を述べた．提案方式は，重

© 2017 Information Processing Society of Japan

要サービスの関連ファイルを別仮想計算機に配置し，重要サービスからのみ参照可能とする．これにより，攻撃者が重要ファイルの存在から重要サービスの存在を特定することを困難するだけでなく，重要ファイルを仮想ディスク単位で隔離しているため，重要ファイルの意図しない書き換えを防止できる．残された課題として，提案方式の評価がある．

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 Trans. Inf. Forensic Secur.*, Vol. 7, No. 5, pp. 1439–1447 (2012).

[2] Min, B., Varadharajan, V., Tupakula, U. and Hitchens, M.: Antivirus security: naked during updates, *Software: Practice and Experience*, Vol. 44, No. 10, pp. 1201–1222 (2014).

[3] Grégio, A. R. A., Afonso, V. M., Filho, D. S. F., Geus, P. L. d. and Jino, M.: Toward a Taxonomy of Malware Behaviors, *The Computer Journal*, Vol. 58, No. 10, pp. 2758–2777 (2015).

[4] Min, B. and Varadharajan, V.: A novel malware for subversion of self-protection in anti-virus, *Software: Practice and Experience*, Vol. 46, No. 3, pp. 361–379 (2016).

[5] Almeshekah, M. H. and Spafford, E. H.: Planning and integrating deception into computer security defenses, *Proc. 2014 Workshop on New Security Paradigms Workshop*, ACM, pp. 127–138 (2014).

[6] Araujo, F., Hamlen, K. W., Biedermann, S. and Katzenbeisser, S.: From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation, *Proc. 2014 ACM SIGSAC conference on computer and communications security*, ACM, pp. 942–953 (2014).

[7] 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).

[8] Sato, M., Yamauchi, T. and Taniguchi, H.: Memory Access Monitoring and Disguising of Process Information to Avoid Attacks to Essential Services, *2016 Fourth International Symposium on Computing and Networking (CANDAR)*, IEEE, pp. 635–641 (2016).

[9] 佐藤将也，山内利宏，谷口秀夫：攻撃回避のためのファイル不可視化手法の提案，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 2016 (CSS2016) 論文集，Vol. 2016, No. 2, pp. 224–228 (2016).

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

[11] 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).

[12] 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).

[13] Wang, J., Yu, M., Li, B., Qi, Z. and Guan, H.: Hypervisor-based protection of sensitive files in a compromised system, *Proc. 27th Annual ACM Symposium on Applied Computing*, ACM, pp. 1765–1770 (2012).

© 2017 Information Processing Society of Japan