プロセス関連情報の不可視化によりプロセスの識別を困難にする攻撃回避手法

佐藤 将也　　山内 利宏

岡山大学大学院自然科学研究科
700-8530 岡山県岡山市北区津島中 3-1-1
m-sato@swlab.cs.okayama-u.ac.jp　yamauchi@cs.okayama-u.ac.jp

あらまし　ウィルス対策ソフトウェアや管理ツールなど，セキュリティにおいて重要なソフトウェアは，攻撃の対象となり，無力化される恐れがある．そこで，攻撃者から重要なソフトウェアの識別を困難にし，攻撃の対象から除外する手法として，プロセスの識別を困難にする攻撃回避手法を提案する．提案手法では，カーネル内に保持するプロセス関連情報を偽の情報に動的に置換することにより，プロセス関連情報をもとにしたプロセスの識別を困難にする．提案手法を，仮想計算機から独立した仮想計算機モニタの改変により実現することで，提案手法自体の安全性を高める．また，仮想計算機モニタ以外は改変しないことで，既存の OS や AP の改変なしに提案手法を実現する．

Attack Avoiding Method Obscuring Process Identification by Making Process Information Invisible

Masaya Sato　　Toshihiro Yamauchi

Graduate School of Natural Science and Technology, Okayama University
3-1-1 Tsushima-naka, Kita-ku, Okayama, 700-8530 JAPAN

Abstract　Security-critical software has risk of attacks on the software itself to disable its functionality by adversaries. To decrease the risk, we propose an attack avoiding method that obscures process identification. The proposed method obscures process identification based on process-related information by dynamically replacing that information held by a kernel to dummy information. Implementation of the proposed method with a virtual machine monitor enhances security of the mechanism itself. Further, by implementing the proposed method with a virtual machine monitor, modification to OSes and APs is not required.

1　はじめに

プログラムの脆弱性やマルウェアを用いた攻撃が増加している．マルウェアを用いた攻撃を検知し，対処するためのソフトウェアとして，ウィルス対策ソフトウェアが研究開発されている．しかし，このようなソフトウェアは，攻撃者にとって不都合である場合が多いことから，攻撃の対象となり，無力化される恐れがある．例えば，Agobot[1] は，プロセスの実行ファイル名を検査し，ウィルス対策ソフトウェアに利用されているプログラム名が含まれている場合は，そのプロセスを停止する機能を持つ．2013 年 8 月 8 日時点で，Agobot には，停止対象として 579 種類のプログラム名が定義されている．

ウィルス対策ソフトウェアの一つに，侵入検知システム（Intrusion Detection System, IDS）がある．IDS は，計算機のメモリやファイルの

状態を監視し，計算機への攻撃者の侵入を検知するシステムである．ホスト型 IDS への攻撃への対処として，IDS を計算機外部へオフロードする研究 [2, 3, 4] が提案されている．これらの研究は，仮想化技術を用い，IDS を監視対象の仮想計算機（Virtual Machine, 以降，VM）とは異なる VM で動作させる．仮想化技術では，VM 間の強い隔離が提供されているため，オフロードにより IDS は安全に走行できる．しかし，文献 [2] の手法で取得できるのは，VM のメモリ，レジスタ，およびディスクなど，低レイヤの情報のみであり，取得できる情報が既存の IDS と異なる．このため，取得できる情報に合わせて IDS を新たに作成する必要がある．同様に，文献 [3] の手法は，オフロードするために既存 IDS を改変する必要がある．文献 [4] は，システムコールエミュレータをオフロード先の VM で実現することで，既存 IDS を改変なしにオフロードする手法を提案している．しかし，IDS が利用するシステムコール毎に対応する必要があり，実現の工数が大きい．

このように，ウィルス対策ソフトウェアへの攻撃を防止する手法は提案されているものの，それぞれに問題がある．これらの問題への対処として，本稿では，プロセスの隠ぺいにより，重要サービスへの攻撃を困難にする攻撃回避手法を提案する．提案手法では，カーネル内に保持する重要サービスのプロセスに関連する情報（以降，プロセス関連情報）を偽の情報に動的に置換することで，プロセス関連情報をもとにしたプロセスの識別を困難にする．提案手法では，重要サービスの走行中のみ正規のプロセス関連情報を用い，それ以外のプロセスの走行中は，重要サービスのプロセス関連情報を偽の情報に置換する．また，提案手法を仮想計算機モニタ（VM Monitor, 以降，VMM）の改変により実現することで，提案手法自体の安全性を高める．さらに，VMM 以外の改変なしに提案手法を実現することで，既存の OS や AP の改変なしに，重要サービスへの攻撃を回避できる．

2 想定する攻撃

想定する攻撃者は，管理者権限を取得可能であり，カーネル空間とユーザ空間のメモリに自由にアクセスできるものとする．このような攻撃者がプロセスを停止するような攻撃を想定する．管理者権限を取得した攻撃者は，攻撃者に不利益なプロセスを任意に停止できる．例えば，攻撃を検知されないように，マルウェア対策ソフトウェアを停止する攻撃が考えられる．また，ログイン履歴を記録されないようにログ収集プログラムを停止する攻撃が考えられる．

このような攻撃の例として，Agobot[1] がある．1 章で述べたように，Agobot は，ウィルス対策ソフトウェアを停止する機能を持つ．これらのソフトウェアが攻撃により停止した場合，攻撃の検知や対処が遅れることやシステムを管理不能になる問題がある．このため，本稿では，重要サービスとして，ウィルス対策ソフトウェアやシステム管理ツールを想定し，重要サービスへの攻撃の対処を検討する．

3 プロセスの識別を困難にする攻撃回避手法

3.1 目的

本研究の目的は，重要サービスへの攻撃の回避である．攻撃者は，特定のサービスを攻撃する場合，そのサービスが動作していることを検知し，サービスに応じて攻撃する．そこで，重要サービスを提供するプロセスへの攻撃を回避することで，攻撃による被害を抑制する．

3.2 設計方針

（方針 1） 重要サービスの存在を攻撃者から隠ぺいすること

（方針 2） 重要サービスを改変しないこと

重要サービスへの攻撃を回避するには，計算機への侵入防止や重要サービスの計算機外部へのオフロードが有効である．しかし，1 章で述べたような問題がある．そこで，攻撃者が計算機へ侵入したとしても重要サービスへの攻撃を困難にするために，重要サービスの存在を隠ぺいする．重要サービスが隠ぺいされていると，攻撃対象を特定できないため，重要サービスへの攻撃は難しい．

また，重要サービスを改変することで攻撃を回避すると，攻撃を回避するために，サービスを個別に改変する必要がある．個別の改変は，

修正工数が大きくなるため，重要サービスの改変なしに，攻撃を回避できることが望ましい．

なお，2 章の想定より，本稿における検討では，仮想化技術の利用を前提とする．攻撃者がある OS のユーザ空間とカーネル空間に自由にアクセスできる環境では，攻撃へ対処するには，OS から独立した環境が求められる．そこで，VMM を用い，攻撃者から自由にアクセスできない環境において，攻撃に対処する．VMM を用いた詳細な検討は，3.4.2 項で述べる．

3.3 課題

3.1 節の目的を達成する機構を 3.2 節の方針に従い設計する．設計における課題を以下に示す．

（課題 1） 重要サービスを提供するプロセスに関連する情報の不可視化

（課題 2） プロセス関連情報を不可視化する機構自体の存在を検知されないこと

サービスを提供しているプロセスを特定するには，プロセス関連情報が必要である．このため，プロセス関連情報を攻撃者から不可視化することで，重要サービスを提供しているプロセスの存在を隠ぺいする．

また，プロセス関連情報を不可視化する機構を実現したとしても，攻撃者からその機構自体を検知されると，対処される恐れがある．このため，プロセス関連情報を不可視化する機構自体の存在を検知されないことが求められる．

3.4 課題への対処

3.4.1 （課題 1）への対処

（課題 1）への対処として，以下の 2 方式が考えられる．

（方式 1） プロセス関連情報へのアクセス制御

（方式 2） プロセス関連情報の置換

（方式 1）について，図 1 に概略図を示す．プロセス関連情報へのアクセスを許可するカーネルテキスト部の領域を事前に設定しておき，また，プロセス関連情報が存在するページの読み込みを事前に禁止しておく．（方式 2）では，プロセス関連情報へのアクセス違反を検知する機構により，アクセス制御を行う．アクセス違反が発生した際に，アクセス元の命令が配置されているアドレスが事前に許可されたカーネルテキスト部の場合は，アクセスをエミュレートする．そうでない場合は，不正なアクセスとして処理する．

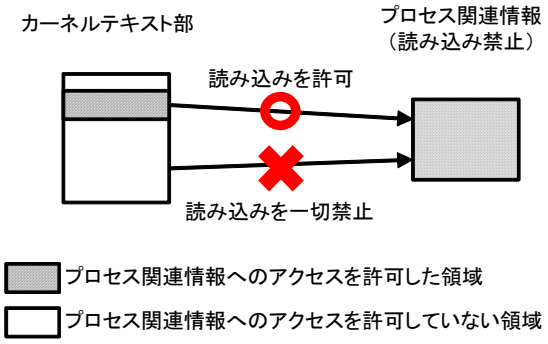


図 1 プロセス関連情報へのアクセス制御

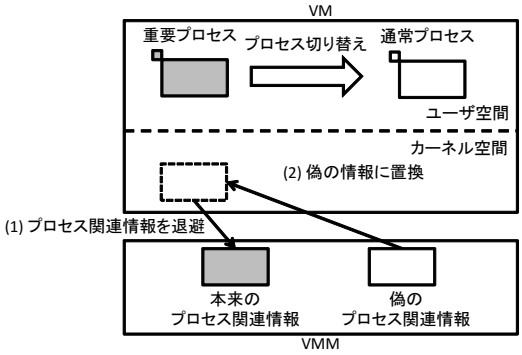


図 2 重要プロセスから通常プロセスへの切り替え時におけるプロセス関連情報の置換

アクセス制御では，プロセス関連情報へのアクセス元の命令が配置されているメモリアドレスをもとに制御する．この方式では，命令が配置されているメモリアドレスがカーネルテキスト内であれば，アクセスを許可し，その他の領域であれば，アクセスを制限する．

また，カーネルスタックの検査により，プロセス関連情報へアクセスした関数の呼び出し元を特定する手法 [5] を用いる必要がある．これは，カーネルに不正なコードを挿入することなく，正規のコードの呼び出すことで，重要サービスのプロセス（以降，重要プロセス）のプロセス関連情報を参照されるのを防ぐためである．これにより，事前に許可したカーネルテキスト部以外からは，重要プロセスのプロセス関連情報の参照を禁止できる．

（方式 2）について，図 2 と図 3 に概略図を示す．（方式 2）では，重要プロセスのプロセス

表 1 各方式の比較

	利点	欠点
(方式 1)	アクセス制御により，すべてのプロセス関連情報を不可視化できる.	● アクセス制御ポリシを事前に設定する困難さがある. ● ページ単位のアクセス制御では，制御の粒度が粗い.
(方式 2)	プロセス関連情報の置換のみで実現できる.	● プロセス関連情報のデータ構造を事前に調査する必要がある. ● 偽のプロセス関連情報を事前に作成する必要がある.

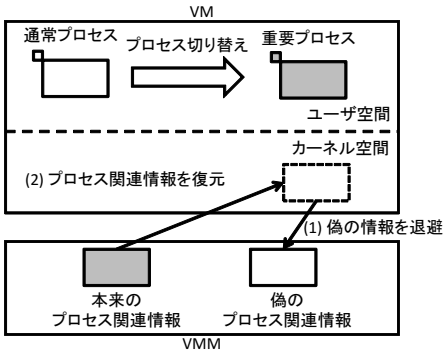


図 3 通常プロセスから重要プロセスへの切り替え時におけるプロセス関連情報の置換

関連情報を事前に偽の情報に置換しておき，重要プロセスの走行中は，本来のプロセス関連情報に復元する．重要プロセスから通常プロセスへのプロセス切り替えが発生すると，図 2 に示すように，重要プロセスのプロセス関連情報を退避し，偽の情報に置換する．また，通常プロセスから重要プロセスへのプロセス切り替えが発生すると，図 3 に示すように，重要プロセスのプロセス関連情報を復元する．重要プロセスから別の重要プロセスに切り替わる場合は，切り替え前の重要プロセスのプロセス関連情報を退避し，偽の情報に置換した後，切り替え先の重要プロセスのプロセス関連情報を復元する．これにより，走行中の重要プロセス以外からは，重要プロセスの正規のプロセス関連情報を参照できない.

それぞれの方式を表 1 で比較する．(方式 1) は，(方式 2) と比べ，すべてのプロセス関連情報を不可視化できる．(方式 2) は，プロセス関連情報を偽の情報に置換することから，偽の情報の設定方法によっては，プロセスを特定される恐れがある．このため，(方式 2) を用いてプロセス関連情報を不可視化する場合は，偽の情報の設定方法が課題となる．

(方式 1) は，プロセス関連情報へアクセスできるカーネルテキスト部やアクセス制御ポリシを事前に設定する必要がある．また，重要プロセスのプロセス関連情報がメモリ上のどこに配置されるのかが既知である必要がある．さらに，アクセス制御の粒度は，ページ単位である．このため，プロセス関連情報へのアクセス違反を検知した後，アクセス先が重要プロセスのプロセス関連情報か否かを常に判断する必要がある．一方，(方式 2) は，プロセス切り替え時にプロセス関連情報を置換した後は，制御を必要としない．ただし，プロセス関連情報を置換するには，事前にプロセス関連情報のデータ構造が既知である必要がある．また，事前に偽のプロセス関連情報を作成する必要がある.

(方式 1) と (方式 2) は，どちらも，3.2 節における設計方針を満たす．しかし，(方式 1) は，事前に設定すべき情報と動的なアクセス制御の複雑さの問題がある．一方，(方式 2) は，プロセス切り替えの発生時にプロセス関連情報を置換するだけで良い．このため，実現の容易さの点で，(方式 2) が優れる．また，(方式 1) は，プロセス関連情報へのアクセスの度にアクセス制御が必要となる一方，(方式 2) は，プロセス切り替えの際にプロセス関連情報を入れ替えるのみである．このため，性能面でも (方式 2) が

優れる．以上より，(方式 2）を採用する．

3.4.2 （課題 2）への対処

攻撃者は，カーネル空間とユーザ空間のメモリに自由にアクセスできると想定している．この想定より，3.2 節で述べたように，カーネルよりも高い権限で動作し，カーネルからのアクセスを制限できる方法として，VMM による機構の実現が考えられる．VMM は，OS カーネルよりも低レイヤで動作するソフトウェアであり，OS カーネルよりも高い権限を持つ．

VM からは，VMM の存在を検知できるものの，VMM 内で行われている処理は追跡できない．このため，VMM 内でプロセス関連情報を不可視化する機構を実現した場合，VM から機構を検知するのは難しい．(課題 2）への対処では，VM 上の OS や AP を一切改変することなく，VMM の改変のみにより，プロセス関連情報を不可視化する．

提案手法において，VMM により（方式 2）を実現するには，VM 上のプロセス切り替えを機構から検知できる必要がある．プロセス切り替え時には，CR3 レジスタを書き換え，仮想空間を切り替える．VMM は，VM による CR3 レジスタの操作を検知できることから，VM 上のプロセス切り替えを検知できる．VMM による VM 上のプロセス切り替えの検知については，4.3.1 項で述べる．

3.5 提案手法の処理の流れ

以上の対処を実現した場合の提案手法の処理の流れを図 4 に示す．提案手法では，VM 上のプロセス関連情報の置換を VMM により実現する．VMM は，VM 上のプロセス切り替えを検知すると，切り替え前のプロセスが重要プロセスの場合は，重要プロセスのプロセス関連情報を VMM 内に退避し，偽のプロセス関連情報で置換する．また，切り替え先のプロセスが重要プロセスの場合は，当該重要プロセスのプロセス関連情報を復元する．切り替え前または切り替え先のプロセスが通常プロセスの場合は，処理を行わない．

重要プロセスのプロセス関連情報は，VMM 内のプロセス関連情報退避部に退避する．VMM

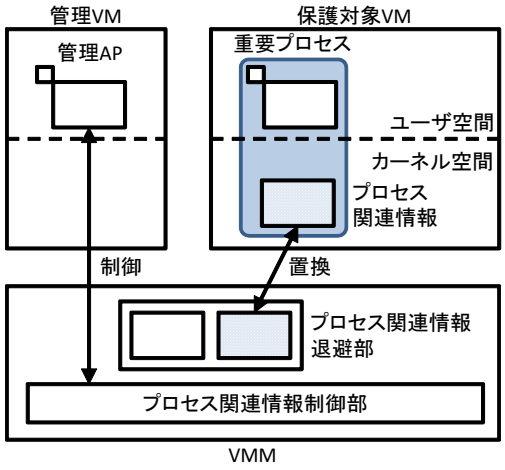


図 4 提案手法の処理の流れ

内の情報は VM からは参照できないため，重要プロセスの実行コンテキスト以外では，重要プロセスのプロセス関連情報を参照できない．

管理 VM 上の管理 AP は，保護対象とする重要プロセスを指定する．VMM 内のプロセス関連情報制御部は，管理 AP からの指示に従い，どのプロセスを重要プロセスとするか決定する．また，プロセス関連情報制御部は，保護対象 VM 上のプロセス切り替えを検知し，重要プロセスのプロセス関連情報を置換する．プロセス関連情報の置換の詳細は，4.3.2 項で述べる．

提案手法を VMM により実現した場合，提案手法の存在を VM から検知できないだけでなく，提案手法への攻撃も困難にできる．また，VM 上の既存ソフトウェアを一切改変することなく，重要サービスへの攻撃を困難にできる．

3.6 提案手法の限界

提案手法は，プロセス関連情報を不可視化し，プロセスの識別を困難にするのみであり，プロセス自体を不可視にはしない．このため，偽のプロセス関連情報に置換した状態のプロセスは，攻撃者から認識できる．また，攻撃者が無作為にプロセスを指定して動作を停止させた場合には，重要プロセスを停止される恐れがある．このような攻撃に対処するには，プロセス自体を不可視化するか，プロセス関連情報の攻撃者による書き換えを防止する必要がある．書き換えを防止する場合には，(方式 1）と（方式 2）を

併用する．

また，LKM などを用いて重要プロセスの実行コンテキストで処理を実行された場合には，重要プロセスのプロセス関連情報は，攻撃者から可視である．LKM を用いた攻撃には，主に，カーネルのテキスト部を書き換える手法，データ部にコードを挿入して実行させる手法，および LKM から正規のカーネルコードを呼び出すことでカーネル内の情報を取得する手法がある．これらの攻撃は，既存手法の利用により対処できる．カーネルのテキスト部やデータ部にコードを挿入して実行する手法は，書き換え可能なメモリに実行権限を付与しない手法 [6, 7] により対処できる．正規のカーネルコードを呼び出すことでカーネル内の情報を取得する手法については，3.4.1 項で（方式 1）として述べたプロセス関連情報へのアクセス制御により対処できる．

4 プロセスの不可視化

4.1 想定する環境

プロセス関連情報の置換を実現するには，4.2 節で示すプロセス関連情報を操作できる必要がある．このため，OS として，ソースコードが公開されている Linux を想定する．VMM としては，Xen を用い，VM は完全仮想化されているものとする．これは，準仮想化では，VM から VMM に対して明示的にアクセスするインタフェースが提供されているため，VMM の脆弱性を突いて，退避したプロセス関連情報を参照される恐れがあるためである．完全仮想化環境では，VM から VMM への処理の遷移は，VM から明示的に遷移させるものではなく，VM 上のイベントを VMM が検知することで処理が遷移する．また，マルチコア CPU を想定し，VM に複数の仮想 CPU コアを割り当て可能とする．

4.2 プロセス関連情報

プロセスの不可視化では，プロセス関連情報を攻撃者から不可視化する．プロセス関連情報とは，そのプロセスの起動により生成され，プロセスが終了するまでに参照や書き換えの対象となるものとする．Linux では，プロセス関連情報として以下がある．

(1) プロセス制御ブロック
(2) カーネルスタック
(3) ハードウェアコンテキスト
(4) ページテーブル
(5) プロセスが利用しているメモリ

プロセスを完全に不可視化するには，上記のプロセス関連情報をすべて不可視化する必要がある．レジスタの内容やプロセスが利用しているメモリをもとにそのプロセスの動作内容を把握するのは困難である．一方，プロセス制御ブロックやカーネルスタックには，プロセスを識別するための情報が多く格納されている．例えば，プロセス制御ブロックには，プロセスの実行ファイルのファイル名が格納されており，マルウェアは，そのプロセスがどのようなプロセスなのかを容易に判定できる．また，カーネルスタックはプロセス毎に作成され，カーネル内の関数の呼び出し履歴を参照できる．このため，本研究では，プロセス制御ブロックとカーネルスタックを不可視化の対象とする．

4.3 不可視化手法

4.3.1 プロセス切り替えの検知

VMM からプロセス VM 上のプロセス切り替えを検知するには，CR3 レジスタの書き換えを検知する．CR3 レジスタは，ページディレクトリのアドレスを格納するレジスタであり，ページディレクトリはプロセスごとに異なる．このため，CR3 レジスタの値から，現在走行中のプロセスを識別できる．Xen では，完全仮想化した VM を提供する場合，VM による CR3 レジスタの書き換えを検知する．CR3 レジスタには，VM から観測できる物理アドレスが格納される．VMM は，疑似物理アドレスから物理アドレスへの変換テーブルを維持するために，VM による CR3 レジスタへのアクセスを検知する．ページディレクトリを切り替えるのは，プロセス切り替えの際であることから，提案手法では，CR3 レジスタへのアクセスをプロセス切り替えとして検知する．

ただし，HAP (Hardware Assisted Paging) が有効な環境では，この方法は利用できない．これは，上記のアドレス変換テーブルを CPU が維持するためである．このため，提案手法では，HAP を無効にした環境で実現する．HAP

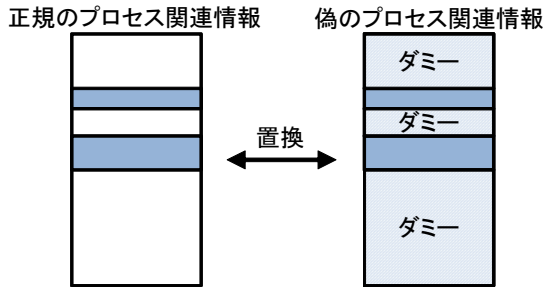


図 5　正規のプロセス制御ブロックの偽のプロセス制御ブロックとの置換

が有効な環境でプロセス切り替えを検知するには，ゲスト OS におけるプロセス切り替え処理にブレークポイントを設定するなどして，プロセス切り替え処理の呼び出しを検知する必要がある．

4.3.2　プロセス関連情報の置換

不可視化のために重要プロセスのプロセス関連情報を置換するには，偽のプロセス関連情報の作成とプロセス関連情報の退避が必要となる．4.2 節で述べたように，置換する対象のプロセス関連情報は，プロセス制御ブロックとカーネルスタックである．このうち，プロセス制御ブロックは，プロセスのスケジューリングに関する情報など，プロセスの走行待機中にカーネルから参照される情報を格納している．カーネルがこれらの情報を参照できないと，重要プロセスが関係するプロセス切り替えで問題が起こる．このため，偽のプロセス制御ブロックは，プロセス制御ブロックすべてを置換するのではなく，プロセスの走行待機中にカーネルから参照される情報として正規の情報を格納し，それ以外の情報を偽の情報で置換する．

プロセス関連情報の退避には，VMM を用いる．VMM は，VM から独立しており，VM 上で動作するプログラムからの攻撃は難しい．このため，偽のプロセス関連情報は，VMM が保持し，4.3.1 項で述べた方法でプロセス切り替えを検知した際に，動的に正規のプロセス関連情報と置換する．図 5 に正規のプロセス関連情報と偽のプロセス関連情報を切り替える様子を示す．図中でダミーと示してある部分は偽の情報であり，正規のプロセスに見せかけるための情報を格納する．プロセスの走行待機中にカーネルから参照される情報は，情報を置換しない．これにより，プロセスの走行を妨害することなく，重要プロセスを偽のプロセスに見せかけることで，重要プロセスの識別を困難にする．

4.3.3　マルチコア CPU への対処

マルチコア CPU を用いる環境を想定した場合，あるコアで重要プロセスの走行中に，別のコアで他のプロセスが同時に走行する状況が考えられる．この場合，重要プロセスのプロセス関連情報は復元されており，本来の情報を利用している．このため，別のコアのプロセスやカーネルから重要プロセスのプロセス関連情報を参照できてしまう．

この問題へ対処するために，重要プロセスの走行中は，通常プロセスが並列に走行することを禁止する．重要プロセスの走行中は，重要プロセスが走行している CPU コア以外のコアを停止する．これは，VM に提供している仮想 CPU を VMM により一時的に停止することで実現する．これにより，通常プロセスによる保護対象プロセスのプロセス関連情報の参照を制限する．

5　関連研究

文献 [8] では，同時動作中の他プロセスからのメモリ読み出し攻撃に対処する暗号化メモリシステムを提案している．このシステムは，シングルコア環境では，マルチタスク OS においても，実際には複数プロセスが同時に実行されるわけではないことに着目し，対象のプロセスの走行中のみ，そのプロセスのメモリを復号する．対象のプロセス以外のプロセスが走行している間はメモリを暗号化することで，他プロセスから対象のプロセスのメモリの解読を困難にする．復号鍵は，カーネルが保持する．文献 [8] は，カーネルを信頼できるものとして，カーネルが保持するプロセス関連情報は暗号化しない．提案手法は，カーネルを信頼できないものとし，カーネルが管理するプロセス関連情報を他プロセスの走行中は不可視化し，重要プロセスの走行時に復元する．

文献 [8] の手法は，プロセスのメモリの解読

は困難である一方，メモリの上書きによる破壊が可能であり，これは，プロセスの動作に問題を生じる．文献 [8] は，カーネル空間への攻撃を想定していないため，対処していない．提案手法は，待機中の重要プロセスのプロセス関連情報を上書きされた場合でも，切り替え先プロセスが重要プロセスの場合にはプロセス関連情報を復元するため，上書きの影響を受けない．

文献 [9, 10] は，プロセスのメモリの不正な書き換えを防止する手法を提案している．攻撃を回避するのではなく，防止するという点で提案手法と目的が異なる．また，攻撃は防止できるものの，プロセスの存在を攻撃者に知られるため，回避策を講じられる可能性がある．提案手法は，重要プロセスがどのようなプロセスなのかの識別を困難にするため，回避策を講じられる可能性が低い．

IDS を計算機外部へオフロードする手法 [3, 4] が提案されている．この手法は，IDS を計算機外部へオフロードすることで，IDS への攻撃を回避している．サービスへの攻撃を回避する点で，提案手法と目的が同じである．しかし，計算機外部へオフロードすることにより，IDS の改変や実現のための工数が大きくなる問題がある．提案手法は，プロセス切り替え時にプロセス関連情報を置換するだけで実現できるため，実現のコストは低い．

6 おわりに

プロセス関連情報の不可視化によりプロセスの識別を困難にする攻撃回避手法を提案した．提案手法は，重要プロセスが走行していない間は，重要プロセスのプロセス関連情報を偽の情報に置換することで，重要プロセスの存在を攻撃者から識別困難にする．これにより，攻撃者は，攻撃対象のプロセスが動作しているか否かを検知できない．また，提案手法を仮想計算機モニタにより実現することで，機構自体の安全性を高め，攻撃者から機構の存在の検知を困難にする．

残された課題として，提案手法の実現，および提案手法を用いた場合の重要プロセスの識別の困難さの評価がある．また，性能への影響を評価する．

謝辞 本研究の一部は，科学研究費補助金特別研究員奨励費による．

参考文献

[1] Agobot, `http://www.f-secure.com/v-descs/agobot.shtml`.

[2] Jiang, X., Wang, X. and Xu, D.: Stealthy Malware Detection Through VMM-Based "Out-of-the-Box" Semantic View Reconstruction, Proc. 14th ACM Conference on Computer and Communications Security (CCS '07), pp.128–138 (2007).

[3] Kourai, K. and Chiba, S.: HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection, Proc. 1st ACM/USENIX International Conference on Virtual Execution Environments (VEE '05). pp.197–207 (2005).

[4] 飯田貴大，光来健一 : VM Shadow: 既存 IDS をオフロードするための実行環境，情報処理学会研究報告，Vol.2011-OS-119, No.6, pp.1–8 (2011).

[5] Srivastava, A. and Giffin, J.: Efficient protection of kernel data structures via object partitioning, Proc. 28th Annual Computer Security Applications Conference (ACSAC '12), pp.429–438 (2012).

[6] Seshadri, A., Luk, M., Qu, N. and Perrig, A.: SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes, Proc. 21st ACM SIGOPS Symposium on Operating Systems Principles, pp.335–350 (2007).

[7] Microsoft: Data Execution Prevention (DEP), `http://support.microsoft.com/kb/875352/EN-US/` (2006).

[8] 稲村 雄，本郷節之 : 暗号技術によるメモリデータ保護方式の提案，情報処理学会論文誌，Vol.45, No.8, pp.1823–1832 (2004).

[9] Dewan, P., Durham, D., Khosravi, H., Long, M., Nagabhushan, G.: A Hypervisor-Based System for Protecting Software Runtime Memory and Persistent Storage, Proc. 2008 Spring Simulation Multiconference (SpringSim '08), pp.828–835 (2008).

[10] McCune, J.M., Yanlin L., Ning Q., Zongwei Z., Datta, A., Gligor, V., Perrig, A.: TrustVisor: Efficient TCB Reduction and Attestation, Proc. 2010 IEEE Symposium on Security and Privacy, pp.143–158 (2010).