V-Met: IaaS型クラウドにおける ネストした仮想化を用いた安全な監視機構

美山 翔平[1]　光来 健一[1]

概要：IaaS 型クラウドはネットワークを介してユーザに仮想マシン（VM）を提供し，ユーザは自由にシステムを構築することができる．その一方で，クラウド内の VM は十分に管理されているとは限らないため，侵入検知システム（IDS）を用いて VM を監視することが重要となっている．IDS を安全に実行するために IDS オフロードと呼ばれる手法が提案されている．しかし，クラウドは十分に信頼できるとは限らないため，クラウド内で IDS オフロードを行っても，IDS が正しく動作していることを保証するのは難しい．これまで，VM の下で動作するハイパーバイザを信頼する手法が提案されてきたが，信頼できるとは限らない一般のクラウド管理者が仮想化システムの一部しか管理できなくなるという問題があった．本稿では，従来の仮想化システム全体を VM 内で動作させることを可能にする技術であるネストした仮想化を用いて，仮想化システムの外側で IDS を動作させ，安全に VM を監視するシステム V-Met を提案する．V-Met では，一般のクラウド管理者に従来の仮想化システム全体の管理権限を与えることができるため，従来通りの管理を行うことが可能となる．その一方で，IDS は従来の仮想化システムの外側の安全な領域で動作するため，一般のクラウド管理者は IDS を攻撃することができない．我々は V-Met を Xen 上に実装し，いくつかの IDS の動作を確認した．

1. はじめに

近年，急速に普及している IaaS 型クラウドはネットワークを介してユーザに仮想マシン（VM）を提供し，ユーザは自由にシステムを構築することができる．その一方で，クラウド内の VM は十分に管理されているとは限らず，外部から侵入されて機密情報が漏洩する危険がある．そのため，侵入検知システム（IDS）を用いて VM を監視することが重要となっている．VM 内で動作させた IDS が侵入時に無効化されてしまうのを防ぐために，IDS を安全に実行するための IDS オフロードと呼ばれる手法が提案されている [1]．この手法は IDS を別の VM 上で動作させ，監視対象 VM の外から監視を行うことを可能にする．これにより，IDS が攻撃を検知する前に攻撃者によって無効化されることを防ぐことができる．

クラウドにおいては，管理者が十分に信頼できるとは限らない [2] [3] [4] [5] [6] ため，IDS オフロードを行っても IDS が正しく動作していることを保証するのが難しい．クラウド管理者に悪意があった場合，IDS が正しく動作しないように改ざんされたり，IDS を無効化されたりする危険性がある．悪意はなくとも，クラウド管理者が十分なセキュリティ対策を行っていない場合，外部から IDS が攻撃を受ける可能性も考えられる．従来，クラウド上で安全に IDS オフロードを行うために，信頼できるハイパーバイザを用いる手法が提案されてきた [7] [8]．しかし，これらの手法には信頼できるとは限らない一般のクラウド管理者が仮想化システムの一部しか管理できなくなるという問題があった．

そこで本稿では， ネストした仮想化 [9] を用いて仮想化システムの外側で IDS を動作させ，安全に VM を監視するシステム V-Met を提案する．V-Met では，一般のクラウド管理者が従来通りに仮想化システム全体の管理を行うことができる．その一方で，IDS が仮想化システムの外側の信頼できる領域で動作するため，仮想化システムの管理権限だけを持った一般のクラウド管理者が IDS を無力化するのは難しくなる．オフロードされた IDS は仮想化システム内の VM のメモリ，ディスク，ネットワークから情報を取得し，VM への侵入を検知する．V-Met 上で Transcall [10] を動作させることで，既存の IDS を実行することも可能となる．

我々は V-Met を Xen 4.4 のハイパーバイザに実装した．このハイパーバイザ上の VM 内で従来の仮想化システムを動作させ，管理 VM 内で IDS を動作させる．この仮想化システム内の監視対象 VM のメモリ情報を取得する際に

[1] 九州工業大学
Kyushu Institute of Technology

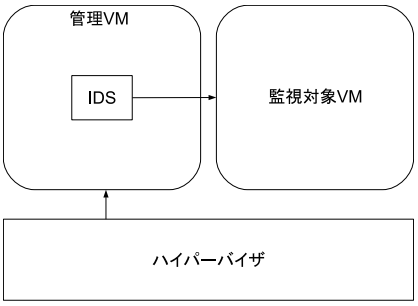

図 1 IDS オフロード

は，監視対象 VM のページテーブルと拡張ページテーブル (EPT) を用いてアドレス変換を行う．ページテーブルのアドレスは，ページテーブルを切り替える際に VM Exit を発生させて取得し，同時に VMCS から EPT のアドレスも取得する．V-Met と Transcall を用いて chkrootkit および Tripwire をオフロード実行したところ，従来の IDS オフロードと比較したオーバヘッドは最大でも 20%程度であることが分かった．

以下，2 章でクラウドにおける IDS オフロードについて詳説し，3 章ではそれらの問題点を踏まえて提案システムである V-Met について述べる．4 章では V-Met の実装について述べ，5 章で V-Met と従来手法との比較のために行った実験の結果を示し，それらについて考察する．6 章では関連研究について述べ，7 章で本稿をまとめる．

2. クラウドにおける IDS オフロードの問題点

IDS を安全に動作させるために VM を用いた IDS オフロード手法が提案されている [1]．この手法は，図 1 のように，監視対象システムが動作している VM と同じホスト上の別の VM で IDS を動作させ，システムの外側から監視を行う．IDS を動作させる VM としては，VM の管理を行う特権を持った管理 VM が用られることが多い．この手法を用いることにより，監視対象 VM が攻撃を受けたとしてもその中で IDS が動作していないため，IDS を攻撃される恐れはない．一方，オフロード先の管理 VM では IDS 以外のサービスをできるだけ動作させないようにすることで攻撃を受けにくくすることができる．

オフロードされた IDS は監視対象 VM の仮想ハードウェアから情報を取得することで監視を行う．例えば，監視対象 VM 内で悪意のあるプロセスが動いていないかどうかを監視するには，カーネルメモリを解析してプロセス情報を取得し，プロセスの名前や所有者などをチェックしたり，プロセスのメモリを調べたりする．また，監視対象 VM のファイルが改ざんされていないかどうかを監視するには，仮想ディスク上のファイルシステムを解析し，ファイルの属性や内容のチェックを行う．ネットワーク経由の不正アクセスを監視するには，仮想 NIC からパケットを取得して攻撃を検出する．

クラウドおいて IDS オフロードを行うにあたって問題となるのは，クラウドの管理者が常に信頼できるとは限らない [2] [3] [4] [5] [6] ということである．クラウド管理者に悪意があった場合，管理 VM で容易に不正アクセスを行うことができる．また，クラウド上の VM はマイグレーションで移動することがあり，セキュリティ意識の低い管理者やスキルの低い管理者のいるデータセンタで VM が動作する可能性もある．管理 VM に脆弱性がある場合，外部からの攻撃者によって管理 VM の制御が奪われる恐れがある．

そのため，オフロードされた IDS が安全に動作することを担保することができない．悪意を持った管理者や管理 VM に侵入した攻撃者は，IDS を停止させることで侵入検知を回避することができる．また，IDS を改ざんすることで，攻撃を検出しないようにすることもできる．IDS を改ざんしなくとも，IDS が監視対象 VM の仮想ハードウェアを参照する際に，参照先を変更するだけでも IDS の挙動を変えて無力化することができる．

クラウド上で安全に IDS オフロードを行うために，ハイパーバイザを信頼する手法が提案されてきた．例えば，Self-Service Cloud (SSC) [7] では，ユーザはサービスドメインと呼ばれる VM を安全に起動することができ，その中に IDS をオフロードすることができる．クラウドの管理者であってもサービスドメインに干渉することはできない．RemoteTrans [8] はクラウドの外部に IDS をオフロードし，クラウド内のハイパーバイザ経由で監視対象 VM を監視する．ユーザが管理している信頼できるホスト上で IDS を動作させることによって，IDS が停止されたり改ざんされたりすることを防ぐことができる．

しかし，このような従来の手法では，一般のクラウド管理者がハイパーバイザを管理できなくなるという問題がある．一般のクラウド管理者を信頼せず，少数の信頼できるクラウド管理者がハイパーバイザを管理することを前提としているためである．一般的に，ハイパーバイザのアップデートは OS のパッケージを用いて行われるが，依存関係のためにハイパーバイザを含めた仮想化ソフトウェア全体を同時にアップデートする必要がある．そのため，仮想化ソフトウェアのアップデートは信頼できる管理者しか実行できないということになり，一般のクラウド管理者の権限を大幅に制限しなければならなくなる．

3. V-Met

本稿では，ネストした仮想化 [9] を用いて仮想化システムの外側で IDS を動作させ，安全に VM を監視するシステム V-Met を提案する．図 2 に V-Met のシステム構成を示す．V-Met では，一般のクラウド管理者にハイパーバイザを含めた仮想化システム全体を管理する権限を与えることができるため，一般の管理者が従来通りの管理を行うことが可能となる．その一方で，仮想化システムの外側にあ

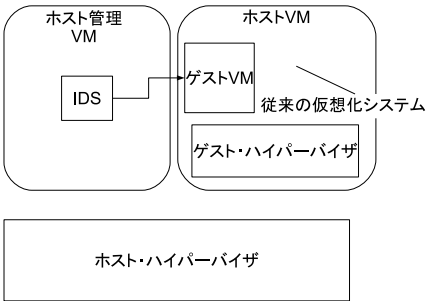

図 2 V-Met のシステム構成

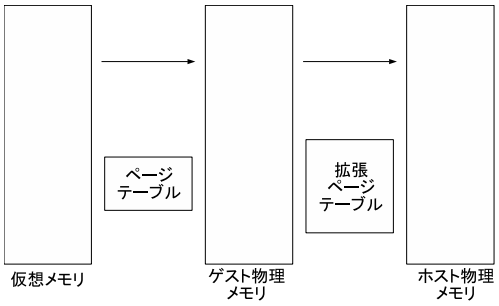

図 3 ゲスト VM にアクセスするためのアドレス変換

る IDS から仮想化システム内の VM を監視するため，仮想化システムの管理権限だけを持った一般のクラウド管理者が IDS を攻撃するのは難しい．

ネストした仮想化は，従来の仮想化システム全体を VM 内で動作させることを可能にする技術である．本稿では，区別のために，従来の仮想化システムにおけるハイパーバイザ，VM をそれぞれゲスト・ハイパーバイザ，ゲスト VM と呼び，この仮想化システムを動作させるためのハイパーバイザ，VM をそれぞれホスト・ハイパーバイザ，ホスト VM と呼ぶ．V-Met では，ホスト・ハイパーバイザ上で動くホスト管理 VM 内でオフロードした IDS を動作させる．ホスト・ハイパーバイザおよびホスト管理 VM は信頼できるクラウド管理者が管理し，ホスト VM 内の仮想化システムは信頼できない一般のクラウド管理者が管理する．ネストした仮想化によるオーバヘッドを削減するために様々な手法が提案されている [2]．これらの手法を用いることで性能低下を 6〜8%程度に抑えることが可能である．また，TinyChecker [11] のような軽量なホスト・ハイパーバイザを用いれば性能低下を 1%程度に抑えることも可能である．

V-Met では，ホスト管理 VM 上の IDS がゲスト VM のメモリ上のデータにアクセスするために，図 3 のように 2 回のアドレス変換を行う．まず，ゲスト VM 内のページテーブルを用いて，監視対象データの仮想アドレスをゲスト VM における物理アドレス（ゲスト物理アドレス）に変換する．次に，ゲスト・ハイパーバイザ内の拡張ページテーブル（EPT）を用いて，ゲスト物理アドレスをホスト VM における物理アドレス（ホスト物理アドレス）に変換する．ゲスト・ハイパーバイザのメモリを解析することでこれらのテーブルのアドレスを取得することができるが，ゲスト・ハイパーバイザは信頼できないため，CPU の仮想化支援機構を利用する．具体的には，ホスト・ハイパーバイザがゲスト VM によるページテーブルの切り替えをトラップし，その際に EPT も特定する．

V-Met では，IDS がゲスト VM の仮想ディスクにアクセスできるようにするために，ネットワークストレージを用いてディスクイメージを共有する．クラウドにおいてゲスト VM はマイグレーションされるため，これは従来と同様のシステム構成である．一方，ゲスト VM が送受信するネットワークパケットはホスト VM の仮想 NIC から取得する．ゲスト VM のパケットはゲスト・ハイパーバイザを経由してホスト・ハイパーバイザに送られるので，ホスト管理 VM において取得可能である．ただし，ホスト VM が送受信するすべてのパケットが含まれるので，ゲスト VM のパケットだけを抽出する．

V-Met 上で Transcall [10] を動作させることで，ホスト管理 VM 上で既存の IDS を実行することができる．Transcall は，既存の IDS をオフロードするための実行環境である VM Shadow を提供するシステムである．Transcall はシステムコール・エミュレータと Shadow ファイルシステムで構成される．システムコール・エミュレータによって VM Shadow の中で動作する IDS が発行するシステムコールをエミュレートし，ゲスト VM のカーネル情報を返す．Shadow ファイルシステムはゲスト VM で使われているものと同じファイルシステムを提供する．特に，特殊なファイルシステムとして Shadow proc ファイルシステムを提供する．Shadow proc ファイルシステムはゲスト VM のカーネルの状態や実行中のプロセス情報などを含んでいる．

4. 実装

我々は V-Met を Xen 4.4 に実装した．V-Met を実装したハイパーバイザをホスト・ハイパーバイザとして動作させ，ホスト VM 内では既存のハイパーバイザをゲスト・ハイパーバイザとして動作させる．ホスト VM およびゲスト VM はどちらも Intel VT-x を用いて完全仮想化で動作させる．

4.1 メモリ監視

ホスト管理 VM からゲスト VM のメモリ上のデータにアクセスするには，データの仮想アドレスをホスト物理アドレスに変換し，そのアドレスを含むメモリページをマップする．そのために，仮想アドレスをまずゲスト物理アドレスに変換し，それをさらにホスト物理アドレスに変換する．前者のアドレス変換はゲスト VM 内のページテーブルを用いて行い，後者のアドレス変換はゲスト・ハイパーバイザ内の EPT を用いて行う．

ゲスト VM 内のページテーブルを用いてアドレス変換を

行うために，V-Met は CPU の CR3 レジスタに格納されているページディレクトリのアドレスを取得する．ゲスト VM の仮想 CPU の CR3 レジスタはゲスト・ハイパーバイザが管理しているが，信頼できないゲスト・ハイパーバイザから取得した情報を利用することはできない．そこで，ゲスト・ハイパーバイザに依存せずに CR3 レジスタの値を取得するために，V-Met では図 4 のように，ゲスト VM が CR3 レジスタの値を変更しようとした時にホスト・ハイパーバイザに対して VM Exit を発生させる．従来の実装では，CR3 レジスタへの書き込みが行われても VM Exit は発生しなかった．そこで，コントロールレジスタの読み書きで VM Exit が発生するように，ホスト・ハイパーバイザにおいてホスト VM の VMCS の VM 実行制御フィールドに設定を行う．ホスト・ハイパーバイザでは CR3 レジスタに対する書き込みかどうかを判定し，そうであれば書き込み元のレジスタの値を保存する．ホスト管理 VM 上の IDS は新たに追加したハイパーコールを用いてホスト・ハイパーバイザを呼び出し，保存されている最新の CR3 レジスタの値を取得する．

ゲスト・ハイパーバイザ内の EPT を用いてアドレス変換を行うために，V-Met は VMCS の VM 実行制御フィールドに格納されている EPT ポインタのアドレスを取得する．V-Met では，ゲスト VM が CR3 レジスタへの書き込みによる VM Exit によってホスト・ハイパーバイザに制御を移した時に，VMCS のホスト物理アドレスを保存する．IDS が新たに追加したハイパーコールを用いてホスト・ハイパーバイザを呼び出した時に，保存されている最新の VMCS から EPT のアドレスを取得する．このハイパーコールはゲスト物理アドレスを引数として受け取り，EPT をたどることでアドレス変換を行い，ホスト物理アドレスを返す．このアドレス変換はホスト・ハイパーバイザ内の機能を利用して行う．

信頼できないゲスト・ハイパーバイザ上で動作するゲスト VM 内のページテーブルは，CloudVisor [4] のメモリ隔離技術を用いて保護する．CloudVisor はゲスト VM のメモリへのアクセスを制限するため，ゲスト・ハイパーバイザやゲスト管理 VM がゲスト VM のページテーブルを改ざんすることはできない．同様に，ゲスト・ハイパーバイザ内の EPT も CloudVisor のメモリ所有者追跡技術を用いて保護する．CloudVisor はゲスト VM のメモリだけを EPT に登録可能としているため，ゲスト・ハイパーバイザが EPT を改ざんするのは難しい．

4.2 ディスク監視

ホスト管理 VM はゲスト VM のディスクイメージが置かれたディレクトリを NFS マウントし，そのディスクイメージをループバック・マウントする．その際には，ホスト管理 VM の OS 内のファイルシステムが用いられる．IDS はディスクイメージがマウントされたディレクトリを参照することで，ゲスト VM のファイルシステムにアクセスする．一方，ゲスト管理 VM でも同様に NFS マウントを行い，マウントしたディレクトリ上のディスクイメージを用いてゲスト VM を起動する．

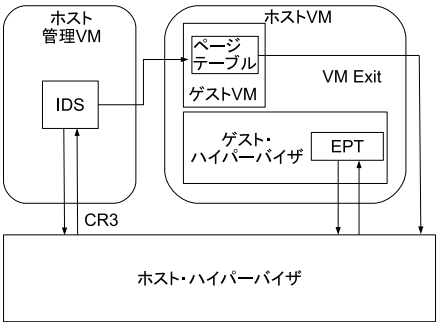

図 4 アドレス変換の流れ

4.3 ネットワーク監視

IDS はホスト管理 VM に作られたホスト VM の仮想ネットワークインタフェース（vif）を通して，ゲスト VM のネットワークパケットを取得する．この vif からはホスト VM が送受信するすべてのパケットが取得されるため，IP アドレスまたは MAC アドレスでフィルタリングすることで，特定のゲスト VM のパケットだけを取得する．ゲスト VM のパケットだけを取得できる仮想ネットワークインタフェースの作成は今後の課題である．

4.4 Transcall の移植

我々は Transcall [10] を V-Met 上に移植した．既存の Transcall は VM の仮想 CPU の状態を取得するハイパーコールを呼び出して CR3 レジスタの値を取得していた．そこで，ゲスト VM の CR3 レジスタの値を取得できるようにするために，ホスト・ハイパーバイザが保存したゲスト VM の CR3 レジスタの値を取得するハイパーコールを呼び出すように変更した．また，既存の Transcall では，ページテーブルエントリに格納された物理ページフレーム番号に対応するページをマップしながら VM のページテーブルをたどっていた．しかし，ゲスト VM のページテーブルエントリにはゲスト物理ページフレーム番号が格納されているため，ハイパーコールを呼び出してホスト物理アドレスに変換してから対応するページをマップするように変更した．

5. 実験

V-Met を用いてオフロードした IDS の動作を確認し，監視性能を調べるための実験を行った．実験には，Intel Xeon E3-1270v3 の CPU，16GB のメモリ，1.8TB の HDD，ギガビットイーサネットを搭載したマシンを用いた．このマ

シンでは，V-Met を実装した Xen 4.4 を動作させ，ホスト管理 VM 上では Linux 3.13.0 を動作させた．ホスト VM では，既存の Xen 4.4 を動作させ，ゲスト管理 VM では Linux 3.13.0 を動作させた．また，ゲスト VM では Linux 3.13.0 を動作させた．比較のために，ネストした仮想化を用いない従来システムでも実験を行い、V-Met におけるホスト VM 内の仮想化システムと同じものを動作させた．NFS サーバには，Intel Xeon X5640 の CPU，32GB のメモリ，16TB の RAID 5 ディスク，ギガビットイーサネットを搭載したマシンを用いた．これらのマシンはギガビットスイッチで接続した．

5.1 オフロードした IDS の動作確認

V-Met と Transcall を用いて chkrootkit をホスト管理 VM で動作させた．chkrootkit はインストールされたルートキットを検知する IDS である．実行結果を比較するために，従来システムにおいて管理 VM に chkrootkit をオフロードして実行を行った．その結果，chkrootkit の実行結果は一致することが分かった．

次に、Tripwire についても同様の実験を行った。Tripwire はファイルシステムの整合性を検査する IDS である。その結果、V-Met における IDS オフロードと従来システムの IDS オフロードにおいて、Tripwire は同じ個数のオブジェクトを検査した。

5.2 メモリ監視性能

V-Met におけるゲスト VM のメモリ監視性能を調べるために，ゲスト VM のカーネルメモリをホスト管理 VM にマップする際のスループットを測定した．比較のために，従来システムにおいて管理 VM から監視対象 VM のカーネルメモリをマップする際のスループットも測定した．図 5 にそれぞれのスループットを示す．結果より，V-Met におけるスループットは従来システムの 40%程度になることが分かった．

次に，カーネルの仮想アドレスをホスト物理アドレスに変換し，対応するページをマップするのにかかる時間の内訳を調べた．図 6 に V-Met と従来システムにおける内訳を示す．CR3 レジスタの値を取得するハイパーコールにかかる時間は処理時間全体の約 2%であり，EPT を用いたアドレス変換を行うハイパーコールにかかる時間は処理時間全体の約 11%であった．よって，V-Met におけるハイパーコールの実行に要する時間は処理時間全体から考えると小さいことが分かった。そのため、V-Met のメモリ監視性能が大幅に低いのには別の原因があると考えられる．

5.3 Shadow proc ファイルシステムの構築時間

Transcall はあらかじめ VM 内のメモリから OS の情報を取得して Shadow proc ファイルシステムを構築する．そこで，Shadow proc ファイルシステムの構築にかかる時間を V-Met と従来システムとで比較した．結果は図 7 のようになり，V-Met は従来システムと比べて約 2.4 倍程度の時間がかかることが分かった．この原因は現在調査中である。

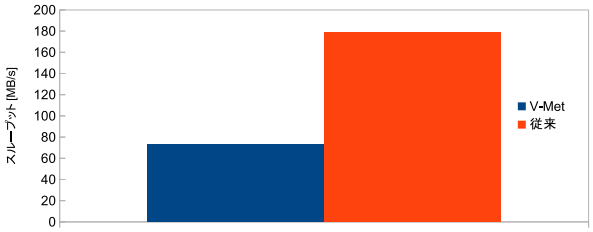

図 5 メモリ監視のスループット

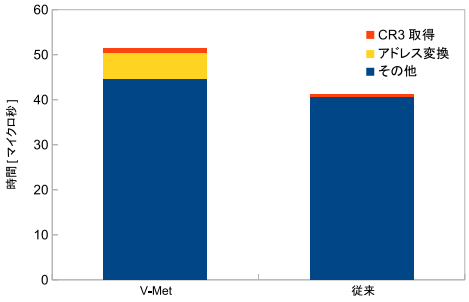

図 6 メモリ監視処理の内訳

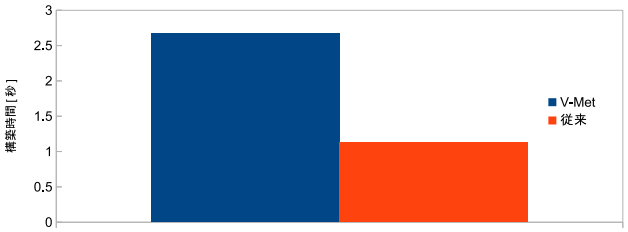

図 7 Shadow proc ファイルシステムの構築時間の比較

5.4 オフロードした IDS の実行時間

V-Met と Transcall を用いてホスト管理 VM で chkrootkit を実行し，ゲスト VM の検査にかかる時間を測定した．比較のために，従来システムのおいて管理 VM で chkrootkit を実行した場合の時間も測定した．実行結果は図 8 のようになり，V-Met によるオーバヘッドは 20%程度であった．一方，chkrootkit をオフロードせずに監視対象 VM 内で実行した場合の実行時間は図 9 のようになった．V-Met では従来システムの約 2 倍の時間がかかっており，ネストした仮想化のオーバヘッドが大きいことが分かる．

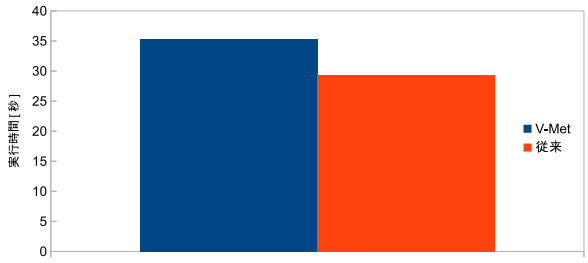

図 8 監視対象 VM 内での chkrootkit の実行時間

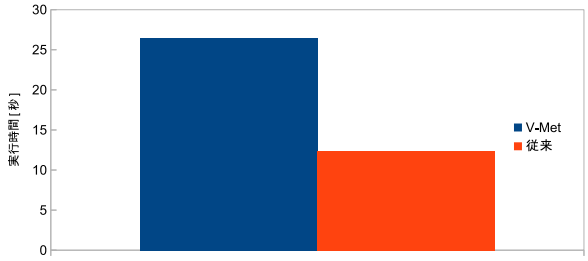

図 9 オフロードした chkrootkit の実行時間

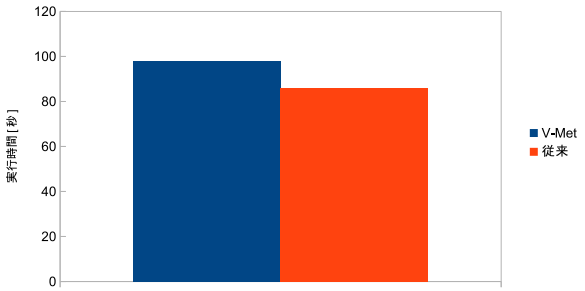

図 10 オフロードした Tripwire の実行時間

次に，Tripwire をオフロードして実行した場合の実行時間を V-Met および従来システムにおいて測定した．実行結果は図 10 のようになり，V-Met では実行時間が 13%長くなることが分かった．一方，監視対象 VM 内で Tripwire を実行した場合，実行時間は図 11 のようになり，V-Met のオーバヘッドは 60%程度であることが分かった．ネストした仮想化により，ゲスト VM から NFS サーバへのアクセス性能が低下することが一つの原因である．

6. 関連研究

Self-Service Cloud (SSC) [7] は，信頼できるハイパーバイザを用いてクラウドのユーザだけに自身の VM を管理する権限を与え，クラウドの管理者からの干渉を防ぐ．ユーザはサービスドメインと呼ばれる VM を安全に起動し，IDS を動作させて他の VM を監視することができる．クラウドの管理者がサービスドメインの中の IDS を停止したり，改ざんしたりすることはできない．しかし，サービスドメイン内のシステムに脆弱性があった場合，攻撃を受ける可能性がある．

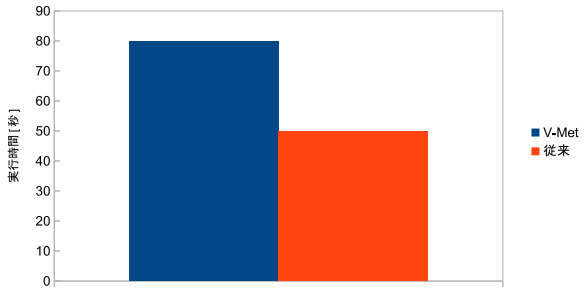

図 11 監視対象 VM 内で実行した Tripwire の実行時間

RemoteTrans [8] は監視対象 VM が動作しているクラウドとは別のホストに IDS をオフロードし，ネットワーク経由で安全に監視対象 VM を監視できるようにするシステムである．リモートホスト上の IDS は信頼できない管理 VM を介して信頼できるハイパーバイザと暗号通信を行い，監視対象 VM のメモリやネットワークパケット，ディスクブロックを取得する．Transcall を用いることにより既存の IDS を動作させることもできる．しかし，IDS をユーザ側で管理する必要があり，IDS を動作させるホストの安全性もユーザが担保する必要がある．

ハードウェアによる安全な実行のサポートを用いることによって，安全に IDS を動作させつつ，一般のクラウド管理者に仮想化システム全体を管理させることが可能である．HyperGuard [12] は CPU のシステムマネジメントモード（SMM）で安全にハイパーバイザのメモリチェックを行う．HyperCheck [13] は SMM を用いてメモリやレジスタの内容をリモートホストに送り，完全性のチェックを行う．HyperSentry [14] は SMM と IPMI を利用してハイパーバイザ内で安全に IDS を動作させる．しかし，SMM で IDS を実行している間はシステムの他の部分が停止してしまうという問題がある．また，SMM での実行は低速なため，IDS の性能に大きな影響を及ぼす．Flicker [15] は Intel TXT や AMD SVM を用いて安全に IDS を実行するが，IDS の実行中は安全性のためにシステムの他の部分を停止させなければならない．

CloudVisor [4] はネストした仮想化を用いてハイパーバイザの下にセキュリティモニタを導入することで，信頼できないクラウドの中で安全に VM を動作させることができる．VM はハイパーバイザと管理 VM から隔離されるため，VM からの情報漏洩を防ぐことができる．しかし，セキュリティモニタから VM を監視する機能は提供されていない．

7. まとめ

本稿では，ネストした仮想化を用いてIDSを仮想化システムの外側で実行するシステムV-Metを提案した．V-Metを用いることで，一般のクラウド管理者が従来通りにハイパーバイザを含めた仮想化システム全体の管理を行うことができるようになる．その一方で，仮想化システムの管理者がIDSを攻撃するのは難しい．我々は既存のIDSを動作させることを可能にするTranscallをV-Metに移植し，いくつかのIDSが正常に動作することを確認した．

今後の課題は，複数のゲストVMが動作している際にそれぞれのゲストVMをホスト管理VMから特定できるようにすることである．また，特定のゲストVMのパケットをだけをIDSに提供する仮想ネットワークインタフェースの作成も行う予定である．

参考文献

[1] T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. *In Proceedings of Network and Distributed Systems Security Symposium*, pp. 191–206, 2003.

[2] C. Li, A. Raghunathan, and N. K. Jha. A Trusted Virtual Machine in an Untrusted Management Environment. *IEEE Transactions on Services Computing, Vol. 5, No. 4*, pp. 472–483, 2012.

[3] C. Li, A. Raghunathan, and N. K. Jha. Secure Virtual Machine Execution under an Untrusted Management OS. *In Proceedings on International Conference on Cloud Computing*, pp. 172–179, 2010.

[4] F. Zhang, J. Chen, H. Chen, and B. Zang. CloudVisor: Retrofitting Protection of Virtual Machines in Multitenant Cloud with Nested Virtualization. *In Proceedings of Symposium on Operating Systems Principles*, pp. 203–216, 2011.

[5] N. Santos, K. P. Gummadi, and R. Rodrigues. Towards Trusted Cloud Computing. *In Proceedings of Workshop on Hot Topics in Cloud Computing*, 2009.

[6] H. Tadokoro, K. Kourai, and S. Chiba. Preventing Information Leakage from Virtual Machines' Memory in IaaS Clouds. *IPSJ Online Transactions, Vol. 5*, pp. 156–166, 2012.

[7] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy. Self-service cloud computing. *In Processings of Conference on Computer and Communications Security*, pp. 253–264, 2012.

[8] 重田一樹, 光来健一. クラウド上の仮想マシンの安全なリモート監視機構. SWoPP 北九州 2013, 2013.

[9] M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour. The Turtles Project: Design and Implementation of Nested Virtualization. In *Proceedings of Symposium on Operating Systems Design and Implementation*, pp. 423–436, 2010.

[10] 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. 第119回OS研究会, 2011.

[11] C. Tan, Y. Xia, H. Chen, and B. Zang. TinyChecker: Transparent Protection of VMs against Hypervisor Failures with Nested Virtualization. *In Proceedings of International Workshop on Dependability of Clouds, Data Centers and Virtual Machine Technology*, 2012.

[12] J. Rutkowska, R. Wojtczuk, and A. Tereshkin. HyperGuard. *Xen 0wning Trilogy, Black Hat USA*, 2008.

[13] J. Wang, A. Stavrou, and A. Ghosh. HyperCheck: A hardware-assisted integrity monitor. In *Proceedings of International Symposium on Recent Advances in Intrusion Detection*, pp. 158–177, 2010.

[14] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C Skalsky. HyperSentry: enabling stealthy in-context measurement of hypervisor integrity. In *Proceedings of Conference on Computer and Communications Security*, pp. 38–49, 2010.

[15] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: An Execution Infrastructure for TCB Minimization. *In Proceedings of European Conference of Computer Systems*, pp. 315–328, 2008.