V-Met：IaaS型クラウドにおける仮想化システム外部からの安全なVM監視

美山 翔平[1]　光来 健一[1]

概要：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，サーバ設定の不備やセキュリティアップデートの未適用など，クラウド内のユーザVMは十分に管理されているとは限らない．そこで，VMの外側で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．しかし，クラウドの管理者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できない．本稿では，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，仮想化システム全体をVM内で動作させるため，仮想化システム内のクラウドの管理者はIDSを攻撃することができない．一方で，クラウドの管理者に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．オフロードしたIDSは仮想化システム内のユーザVMのメモリ，ネットワーク，ディスクから情報を取得し，ユーザVMへの侵入を検知する．実験の結果，従来のIDSオフロードと同程度のオーバヘッドであることが分かった．

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，サーバ設定の不備やセキュリティアップデートの未適用など，クラウド内のユーザVMは十分に管理されているとは限らない．そのため，外部から侵入されて機密情報が漏洩しないように，侵入検知システム（IDS）を用いてユーザVMを監視することが必要となっている．ユーザVM内でIDSを動作させても侵入時に無効化されてしまうため，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きた [4]．この手法はIDSをユーザVMの外側で動作させることにより安全に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．

一方，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 [8] [7] [15] [12] [13] 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を無効化される危険性がある．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，仮想化システム内のハイパーバイザを信頼する手法が提案されてきた [3] [6]．しかし，クラウド管理者は比較的容易にハイパーバイザを攻撃することができる．また，信頼できないかもしれないクラウド管理者には仮想化システムの一部しか管理させられなくなるという問題もあった．

そこで本稿では，仮想化システムの外側から安全にユーザVMを監視するシステムV-Metを提案する．V-Metでは，ネストした仮想化 [2] を用いて従来の仮想化システム全体をVM内で動作させ，その外側でIDSを動作させる．これにより，クラウド管理者に仮想化システム全体の管理権限を与えることができるが，そのような管理者でさえIDSを無効化するのは難しくなる．また，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．オフロードされたIDSは仮想化システム内のユーザVMのメモリ，ディスク，ネットワークから情報を取得し，ユーザVMへの侵入を検知する．V-Met上でTranscall [16] 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のハイパーバイザおよび管理VM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全体を動作させ，管理VM内でIDSを動

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

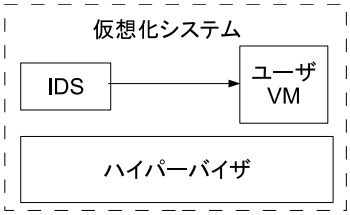

図 1　IDS オフロード

作させる．仮想化システム内のユーザ VM のメモリ上にあるデータを取得するために，V-Met はユーザ 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．ページテーブルと EPT はコンテキストスイッチの際に発生する VM Exit を補足して特定する．また，ユーザ VM が送受信したネットワークパケットはクラウド VM の境界とユーザ VM の境界の二箇所で取得する．ユーザ VM のディスクについては，ネットワーク共有またはクラウド VM の仮想ディスクの解析を通して IDS からアクセスする．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，オーバヘッドは従来の IDS オフロードと同等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ふ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す．6 章で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

IDS を安全に動作させるために VM を用いた IDS オフロードが提案されている [4]．この手法は，図 1 のように，ユーザ VM の外側の仮想化システム内で IDS を動作させ，安全に監視を行うことを可能にする．この手法を用いることにより，ユーザ 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，ユーザ VM の外側の仮想化システム内では IDS 以外のサービスをできるだけ外部に対して提供し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 はユーザ VM から情報を直接取得することで監視を行う．例えば，ユーザ 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する．また，ユーザ 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 を流れるパケットを監視して攻撃を検出する．

2.1 クラウドにおける IDS オフロードの問題

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，管理者が常に信頼できるとは限らない [8] [7] [15] [12] [13] ということである．管理者に悪意があった場合，仮想化システム内で容易に不正アクセスを行うことができる．また，クラウド上のユーザ 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によって管理されている仮想化システム内でユーザ VM が動作する可能性もある．仮想化システムに脆弱性がある場合，外部からの攻撃者によって制御を奪われる恐れもある．

そのため，クラウドではオフロードされた 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仮想化システム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 がユーザ VM を監視する際に，監視先を変更するだけでも IDS の挙動を変えて無効化することができる．

2.2 従来手法

これまでに，クラウド上で安全に IDS オフロードを行うために，仮想化システム内の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) [3] では，ユーザはハイパーバイザ上で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．Remote-Trans [6] はクラウドの外部に IDS をオフロードし，クラウド内で動作するハイパーバイザ経由でユーザ VM を監視する．クラウド管理者はクラウド外部の IDS を停止することはできず，監視に干渉したことは容易に検知される．

しかし，仮想化システム内では管理者は比較的容易にハイパーバイザを攻撃することができる．ハイパーバイザは仮想化システムの管理コンポーネントに様々なインタフェースを提供しており，管理者はそれらのインタフェース経由でハイパーバイザの脆弱性を攻撃できるためである．ハイパーバイザが攻撃されると，IDS を無効化されたり，改ざんされたりする可能性があり，IDS を安全に実行することができなくなる．

また，一般の管理者が仮想化システム全体を管理できなくなるという問題もある．信頼できないかもしれない管理者にハイパーバイザを管理させることはできないためである．一般に，仮想化システムのアップデートはパッケージを用いて行われるが，パッケージ間の依存関係のために仮想化システム全体を同時にアップデートする必要がある．そのため，ハイパーバイザのアップデートだけを別に行えるようにするには，管理手法の大幅な変更が必要となる．

3. V-Met

3.1 仮想化システム外部への IDS オフロード

本稿では，ネストした仮想化 [2] 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．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．本稿ではこの VM をクラウド VM と呼ぶ．図 2 に V-Met のシステム構成を示す．V-Met では，クラウド VM の外側に IDS をオフロードし，IDS はクラウド VM 内のユーザ VM を監視する．クラウドハイパーバイザや IDS はクラウドプロバイダが責任を持って管理し，クラウド VM 内の仮想化システムは信頼できるとは限らない管理者が管理する．

V-Met では以下のような脅威モデルを仮定する．まず，クラウドプロバイダ自身は信頼できるものとする．クラウドプロバイダが信用を失うと致命的であるため，この仮定は広く受け入れられている [8][7][15][12][13][3]．また，クラウドのハードウェアはプロバイダによって適切に管理されているものとし，クラウドハイパーバイザは TPM を用いたリモートアテステーションにより安全に起動することを保証する．クラウドハイパーバイザ上で動作する IDS も安全に実行されるものとする．一方，クラウド VM 内で動作する仮想化システムの管理者は信頼しない．この管理者は仮想化システム内のハイパーバイザや管理コンポーネントに自由にアクセスする権限を持っているが，それ以外の権限は持たないことを想定する．

V-Met はクラウドにおける IDS オフロードの問題を解決することができる．第一に，V-Met では仮想化システムの管理者がその外側で動作している IDS を攻撃するのは難しい．これは，クラウド VM とクラウドハイパーバイザの間のインタフェースは，仮想化システム内の管理コンポーネントとハイパーバイザの間のインタフェースよりも狭いためである．前者のインタフェースは仮想ハードウェアへのインタフェースであり，後者の機能豊富なインタフェースよりも脆弱性が少ない．第二に，信頼できないかもしれない管理者であってもハイパーバイザを含む仮想化システム全体を管理することができる．V-Met では，クラウド VM という仮想化の境界が一般の管理者とクラウドプロバイダの責任分界点となるため，一般の管理者はクラウド VM 内のシステムに自由にアクセスできる．そのため，従来通りの管理を行うことが可能となる．

ネストした仮想化を用いることによりシステムの性能低下が起こるが，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 [3]．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，CloudVisor[15] のセキュリティモニタのように用途を限定したハイパーバイザはネストした仮想化の性能をさらに向上させることができる．近年，ネストした仮想化のハードウェアサポートも追加されている．例えば，VMCS Shadowing[5] は VMCS にアクセスする際の VM Exit を削減することができる．ユーザ VM の監視を行わない間は脱仮想化 [9] によりネストした仮想化のオーバヘッドを削減することができる可能性もある．

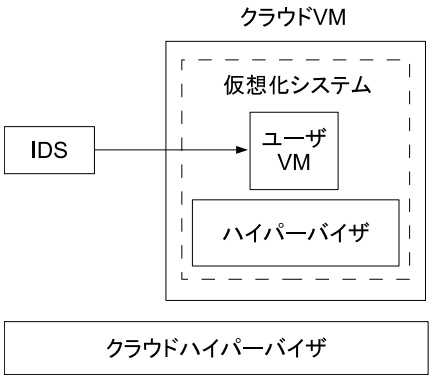

図 2 V-Met のシステム構成

3.2 ユーザ VM の監視

V-Met では，オフロードした IDS はクラウド VM のメモリの中からユーザ VM のデータを見つけて監視を行う．クラウド VM のメモリ上には複数のユーザ VM のメモリが含まれているため，監視対象のユーザ VM のメモリを特定し，さらにその中にある目的のデータを特定する必要がある．そのために，V-Met では 3 回のアドレス変換を行う．まず，ユーザ VM 内のページテーブルを用いて，対象データの仮想アドレスをユーザ VM の物理アドレスに変換する．次に，ユーザ VM 用の拡張ページテーブルを用いて，クラウド VM の物理アドレスに変換する．最後に，クラウド VM 用の拡張ページテーブルを用いてホスト全体の物理アドレスに変換する．

V-Met はユーザ VM が送受信したネットワークパケットをクラウド VM の境界とユーザ VM の境界の二箇所で取得する．クラウド VM の境界で監視を行うことにより，仮想化システムで処理された後の送信パケットや仮想化システムで処理される前の受信パケットを取得することができる．これにより，ユーザ VM が実際に外部と行う通信を調べることができる．一方，ユーザ VM の境界で監視を行うことにより，仮想化システムで処理される前の送信パケットや仮想化システムで処理された後の受信パケットを取得することができる．これにより，ユーザ VM が実際に送受信したパケットそのものを調べることができ，ユーザ VM 間の不正な通信も検知することができる．また，これらの通信ログを突き合わせることにより，仮想化システム内の管理者による攻撃を検出することもできる．

ユーザ VM の仮想ディスクがネットワークストレージ経由で提供されている場合，V-Met は仮想ディスクを IDS と

もネットワーク共有することで監視する．ユーザ VM がマイグレーションされる場合，ネットワークストレージを用いるのが一般的であるため，従来のクラウドと同じシステム構成になる．一方，クラウド VM の中の仮想化システム内にローカルに仮想ディスクが置かれている場合，クラウド VM の仮想ディスクのファイルシステムを解析し，ユーザ VM の仮想ディスクを見つけてアクセスすることで IDS からの監視を行う．

ユーザ VM の ID は仮想化システム内で管理されているため，仮想化システムの外部から安全にユーザ VM を指定することはできない．そこで V-Met では，ユーザ VM の中からウルトラコールを用いてクラウドハイパーバイザに直接，VM を識別するためのタグを登録する．ウルトラコールはユーザ VM が仮想化システムを経由せずにその外側のクラウドハイパーバイザを呼び出すための機構である．そのため，仮想化システムに知られることなく，クラウドハイパーバイザと情報をやりとりすることができる．ユーザはタグを指定して IDS を実行することで，自身のユーザ VM の監視を安全に行うことができる．

V-Met 上で Transcall [16] を動作させることで，既存の IDS をオフロードして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 IDS が発行するシステムコールをエミュレートし，ユーザ VM のカーネル情報を返す．Shadow ファイルシステムはユーザ 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ユーザ VM のカーネルの状態や実行中のプロセス情報などで構成される．

4. 実装

クラウド VM およびユーザ VM はどちらも Intel VT-x を用いて完全仮想化で動作させることを想定している．

4.1 メモリ監視

クラウド管理 VM からユーザ VM のメモリ上のデータにアクセスするために，V-Met は図 3 のように 3 回のアドレス変換を行う．まず，ユーザ VM 内のページテーブルを用いて対象データの仮想アドレスをユーザ VM の物理アドレスに変換する．ユーザ VM 内のページテーブルを用いるためにページディレクトリのアドレスを取得する必要がある．ページディレクトリのアドレスは仮想 CPU の CR3 レジスタに格納されているが，このレジスタは仮想化システム内のハイパーバイザが管理している．しかし，信頼できないハイパーバイザから取得した情報を利用すると安全に監視を行うことはできない．

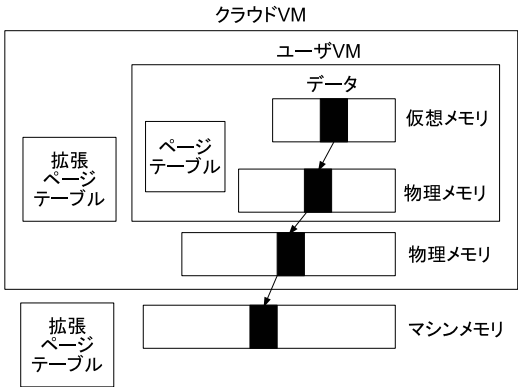

図 3　ユーザ VM のメモリのアドレス変換

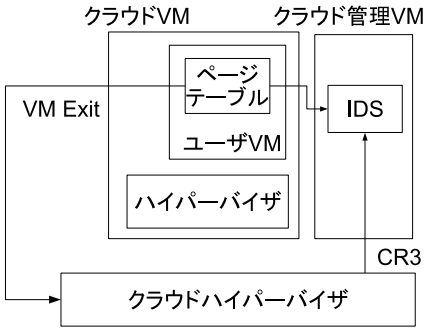

図 4　ユーザ VM の CR3 の取得

そこで，V-Met では図 4 のように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．そのために，ユーザ VM が CR3 レジスタの値を変更しようとした時にクラウドハイパーバイザに対して VM Exit を発生させる．従来は，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，クラウドハイパーバイザにおいてクラウド VM の仮想 CPU 内にある VMCS の VM 実行制御フィールドに設定を行うようにした．クラウド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．クラウド管理 VM 上の IDS は新たに追加したハイパーコールを用いてクラウドハイパーバイザに保存されている最新の CR3 レジスタの値を取得し，ページテーブルをたどってアドレス変換を行う．

次に，ユーザ VM 用の拡張ページテーブル（EPT）を用いて，ユーザ VM の物理アドレスをクラウド VM の物理アドレスに変換する．そのために，仮想化システム内のハイパーバイザが管理している EPT のアドレスを取得する必要がある．V-Met では，図 5 のように，ユーザ VM による CR3 レジスタへの書き込みによって VM Exit が発生した時に，ユーザ VM の仮想 CPU 内にある VMCS から EPT のアドレスを取得する．IDS がハイパーコールを呼び出した時に，ハイパーコール内でこの EPT を用いてアドレス変換を行う．

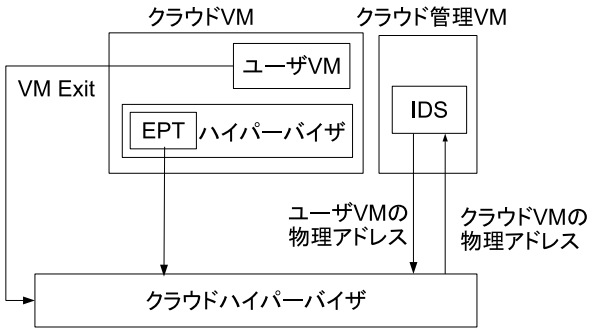

図 5　EPT を用いたアドレス変換

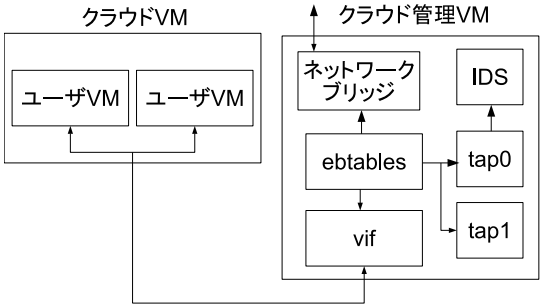

図 6　クラウド VM の境界でのネットワーク監視

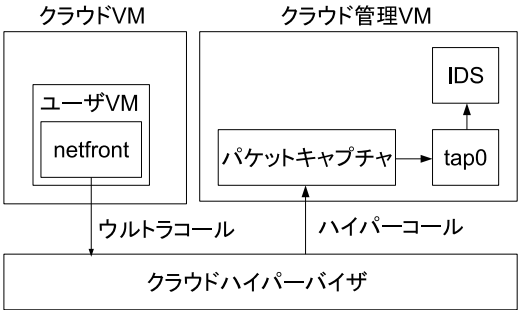

図 7　ユーザ VM の境界でのネットワーク監視

信頼できない仮想化システム上で動作するユーザ VM 内のページテーブルは，CloudVisor [15] のメモリ隔離技術を用いて保護する．CloudVisor は仮想化システムからユーザ VM のメモリへのアクセスを制限するため，ハイパーバイザや管理 VM がユーザ 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，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 はユーザ VM のメモリだけを EPT に登録可能としているため，ハイパーバイザが EPT を自由に改ざんするのは難しい．

最後に，クラウド VM 用の EPT を用いて，クラウド VM の物理アドレスをホスト全体のマシンアドレスに変換する．この EPT はクラウドハイパーバイザ内にあり，クラウド VM のメモリページをマップするハイパーコールを実行する際にこのアドレス変換が自動的に行われる．

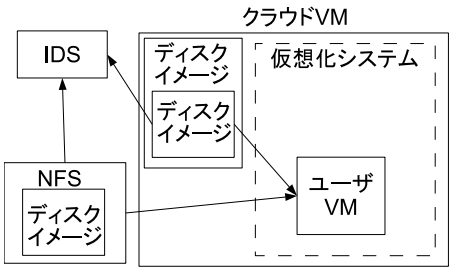

図 8　ディスク監視

4.2　ネットワーク監視

ユーザ VM が送受信したネットワークパケットをクラウド VM の境界で取得する場合，IDS はクラウド管理 VM に作られたクラウド VM 用の仮想 NIC（vif）にアクセスする．この仮想 NIC からはクラウド VM が送受信するすべてのパケットが取得されるため，クラウド VM 内で動作しているすべてのユーザ VM のパケットが混在している．各ユーザ VM のパケットを個別に取得できるようにするために，V-Met では ebtables の ulog 機能を用いる．ulog は ebtables が受信したパケットを netlink ソケットを用いてユーザランドのプログラムに送る機能である．ebtables からパケットを受け取った V-Met は，IDS が監視できるようにパケットを tap デバイスに書き込む．tap デバイスはユーザ VM のそれぞれの MAC アドレスに対して一つ生成する．

そして，ebtables から取得した送信元と宛先のデバイス情報を利用して，送信元デバイスが仮想 NIC の場合には送信元の，宛先デバイスが仮想 NIC の場合には宛先の MAC アドレスに対応する tap デバイスにパケットを書き込む．また，宛先が FF:FF:FF:FF:FF:FF のブロードキャストアドレスの場合や，01:00:5E や 33:33 で始まるマルチキャストアドレスの場合には，すべての tap デバイスにパケットを書き込む．

一方，パケットをユーザ VM の境界で取得する場合には，ユーザ VM のゲスト OS 内のネットワークドライバ（netfront）でパケットを送受信する際にウルトラコールを実行し，クラウドハイパーバイザにパケットのデータを送る．ウルトラコールではデータが格納されているユーザ VM のメモリの物理アドレスが渡されるため，クラウドハイパーバイザ内でユーザ VM 用の EPT を用いてクラウド VM の物理アドレスに変換する．そして，パケットのデータをコピーしてクラウドハイパーバイザ内に保存する．V-Met はクラウド管理 VM で定期的にハイパーコールを呼び出して保存されているパケットのデータを取得し，ユーザ VM ごとに作成された tap デバイスに書き込む．

4.3　ディスク監視

ユーザ VM のディスクイメージがネットワークストレージに置かれている場合，クラウド管理 VM はそのディス

クイメージが置かれたディレクトリを NFS マウントし，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，IDS がユーザ VM が使用中のファイルシステムを破壊しないように，読み込み専用でマウントする．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，ユーザ VM のファイルシステムにアクセスすることができる．仮想化システム内でも同様に NFS マウントを行ったディレクトリ上のディスクイメージを用いてユーザ VM を起動しているため，ディスクイメージを NFS 共有することになる．

一方，クラウド VM の仮想ディスク内にユーザ VM のディスクイメージが置かれている場合，まずクラウド VM のディスクイメージをマウントし，さらにその中のユーザ VM のディスクイメージをマウントする．これらのマウントは読み込み専用で行う必要があるが，マウント時にファイルシステムのリカバリが必要になると問題が生じる．リカバリを行うには，一時的にディスクイメージを書き込み可にする必要があるためである．この問題を解決するために，V-Met では，dm-thin を使ってクラウド VM のディスクイメージのスナップショットを作成する．スナップショットを読み書き可でマウントすることにより，その中のユーザ VM のディスクイメージのマウント時にリカバリを行うことができる．

4.4 ウルトラコール

ウルトラコールは，ユーザ VM が vmcall 命令を発行することにより，直接クラウドハイパーバイザを呼び出す機構である．vmcall 命令はハイパーコールを呼び出す時に用いられる命令である．ネストした仮想化環境において vmcall 命令が実行されると，クラウドハイパーバイザでトラップされた後，通常はクラウド VM 内のハイパーバイザにリダイレクトされる．V-Met ではレジスタに特殊な値が設定されている場合にはリダイレクトを行わず，ウルトラコールとしてクラウドハイパーバイザ内で処理する．

4.5 ユーザ VM の管理

ユーザ VM からクラウドハイパーバイザに登録されたタグは，ユーザ VM 用の EPT と対応づけて管理する．EPT はユーザ VM が起動されるときに作成され，通常は VM の実行中に変更されることはない．さらに，ユーザ VM の CR3 レジスタの値も対応づける．CR3 レジスタはプロセス切り替えのたびに変更される．これらの対応づけは VM の終了時に削除する必要があるが，VM が終了したことの検出は今後の課題である．

4.6 Transcall の移植

既存の IDS をオフロード可能にする Transcall [16] を V-Met 上に移植した．従来の 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．V-Met において，クラウド管理 VM からユーザ VM の CR3 レジスタの値を取得できるようにするために，4.1 節で述べたようにクラウドハイパーバイザで保存された 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，従来の Transcall では VM のページテーブルをたどる際に，ページテーブルエントリに格納された物理アドレスに対応するページをマップしていた．しかし，ユーザ VM のページテーブルエントリにはユーザ VM の物理アドレスが格納されており，クラウド管理 VM で直接マップすることはできない．そのため，ハイパーコールを呼び出してクラウド VM の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 の監視性能やシステム性能に与える影響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 DDR3 SDRAM（1600MHz），2TB の SATA 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，クラウド管理 VM では Linux 3.13.0 を動作させた. クラウド VM には 2 個の仮想 CPU，3GB のメモリ，40GB の仮想ディスクを割り当てた．クラウド VM 内では既存の Xen 4.4 を動作させ，ユーザ VM には 1 個の仮想 CPU，1GB のメモリ，8GB の仮想ディスクを割り当てた．ゲスト管理 VM およびユーザ 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として，V-Met におけるクラウド VM 内の仮想化システムと同じ構成を用いた. NFS サーバには，Intel Xeon X5675 の CPU，32GB のメモリ，3.75TB の RAID5 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 監視性能

V-Met を用いてユーザ VM の監視を行う際の基本的な性能を調べた．V-Met ではクラウド管理 VM からユーザ VM にアクセスする性能を測定した．比較のために，従来システムにおいて管理 VM からユーザ VM にアクセスする性能を測定した．また，ネストした仮想化を用いたシステムにおいてクラウド VM 内のゲスト管理 VM からユーザ VM にアクセスする性能も測定した．

まず，ユーザ VM のメモリ上のデータを読み込むことによりメモリ監視性能を調べた．仮想アドレスを変換してメモリページをマップし，そのメモリの内容をコピーする処理を繰り返してスループットを測定した．実験は結果図 9 のようになり，V-Met は従来システムの 1.3 倍のスルー

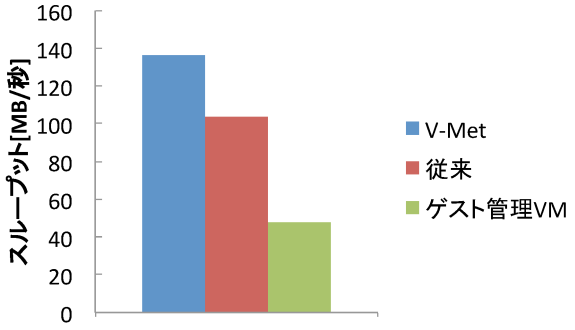

図 9　メモリの読み込み性能

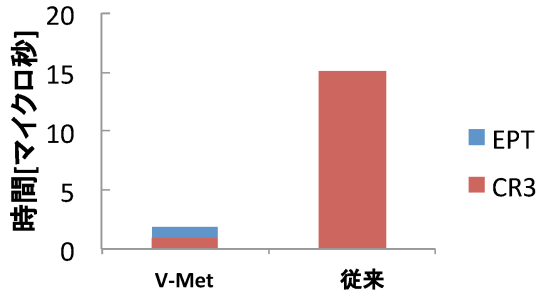

図 10　アドレス変換ハイパーコールの実行時間

プットとなった．これはアドレス変換の実装の差によるものだと考えられる．ゲスト管理 VM でのスループットは，メモリマップのオーバヘッドがより大きくなったため大幅に低下した．

V-Met でのスループットが従来システムより高い理由を調べるために，V-Met においてアドレス変換のために追加したハイパーコールの実行時間を測定した．また，従来システムにおいてアドレス変換の際に使われているハイパーコールについても測定した．その結果は図 10 のようになり，ユーザ VM 用の EPT を用いてユーザ VM の物理アドレスをクラウド VM の物理アドレスに変換するハイパーコールの呼び出しには 1.3μs かかることが分かった．一方，ユーザ VM の CR3 レジスタの値を取得するハイパーコールの呼び出しには 0.96μs かかった．従来システムではユーザ VM の CR3 レジスタの値を取得するのに 15μs かかっていた．これは仮想 CPU の状態を取得する汎用のハイパーコールを用いているためである．

次に，iozone を用いてユーザ VM の仮想ディスクの読み込み性能を調べた．この実験では，あらかじめユーザ VM 内で iozone を実行して 1GB のファイルを作成しておき，それを読み込むスループットを測定した．ファイルキャッシュは実験ごとにクリアして測定を行った．NFS サーバ上に置かれたユーザ VM の仮想ディスクを用いた場合の実験結果は図 11 のようなった．V-Met の性能は従来システムと同程度になった．これは，いずれの場合も仮想ディスクを NFS マウントしてアクセスするためである．ゲスト管理 VM で実行した場合には，クラウド VM のオーバヘッドのために 9%性能が低下した．一方，クラウド VM 内に置かれた仮想ディスクに対して iozone を実行した時の実験結果は図 12 のようになった．NFS を用いた場合とは異なり，V-Met の性能は従来システムと比べて 16%向上した．ゲスト管理 VM での性能も考えると，従来システムでの性能はもっと高いはずであるが，このようにな結果になった原因については究明中である．

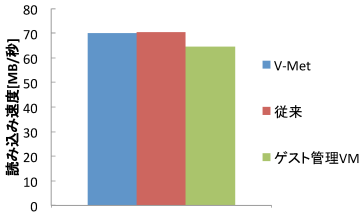

図 11　仮想ディスクの読み込み性能（NFS）

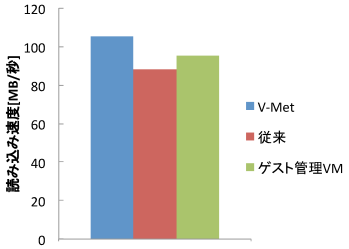

図 12　仮想ディスクの読み込み性能（ローカル）

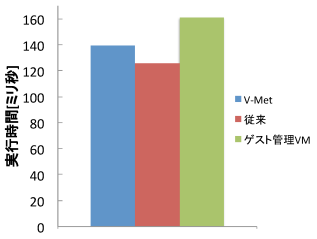

図 13　Shadow proc ファイルシステムの構築時間

5.2　Shadow proc ファイルシステムの構築時間

Transcall の Shadow proc ファイルシステムの構築にかかる時間を測定した．Transcall はユーザ VM 内のメモリからプロセスなどの OS の情報を取得して Shadow proc ファイルシステムを構築する．結果は図 13 のようになり，V-Met によるオーバヘッドは 11%であることが分かった．V-Met では 292 回必要な CR3 レジスタの値を取得するオーバヘッドが削減されたが，EPT を用いたアドレス変換が 3,089 回必要になったためである．

5.3　オフロードした IDS の性能

V-Met を用いてクラウド管理 VM にオフロードした IDS の性能を測定した．比較のために，従来システムにおいて

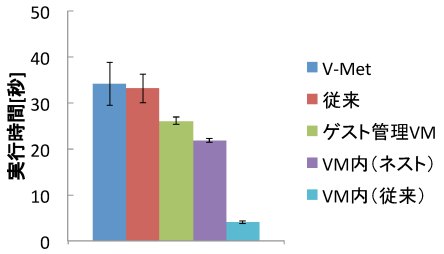

図 14　chkrootkit の実行時間（NFS）

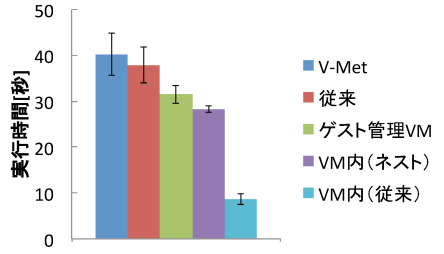

図 15　chkrootkit の実行時間（ローカル）

管理 VM にオフロードした IDS の性能，および，ネストした仮想化を用いたシステムにおいてクラウド VM 内のゲスト管理 VM にオフロードした IDS の性能も測定した．また，ネストした仮想化を用いたシステムと従来システムにおいて，ユーザ VM 内での IDS の性能も測定した．

まず，プロセスやファイルなどを調べてルートキットを検知する IDS である chkrootkit の実行時間を測定した．NFS サーバ上に置かれたユーザ VM の仮想ディスクを用いた場合の実験結果は図 14 のようになった．V-Met における実行時間は従来システムより 3%増加した．一方，ユーザ VM の仮想ディスクをクラウド VM 内に置いた場合の実験結果は図 15 のようになり，V-Met における実行時間は従来システムより 6%増加した．VM 内での実行時間より大幅に増加しているのは，FUSE で実装された Shadow proc ファイルシステムへのアクセスが非常に多く，そのオーバヘッドが顕在化したためである．ゲスト管理 VM での実行時間が従来システムよりも短いが，その原因は調査中である．

次に，ファイルシステムの整合性を検査する IDS である Tripwire の実行時間を測定した．NFS サーバ上に置かれたユーザ VM の仮想ディスクを用いた場合は図 16 のようになった．従来システムと比較すると，V-Met によるオーバーヘッドは 8.7%であった．ゲスト管理 VM での実行時間が短い原因は調査中である．一方，ユーザ VM の仮想ディスクをクラウド VM 内に置いた場合は図 17 のようになり，V-Met における実行時間は従来システムより 13%長くなった．これらの結果より，NFS を用いるほうが全体的に Tripwire の性能がよくなり，ユーザ VM での性能も高いことが分かる．

最後に，nmap コマンドを用いてユーザ VM に対して

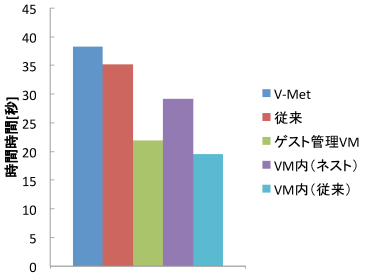

図 16　Tripwire の実行時間（NFS）

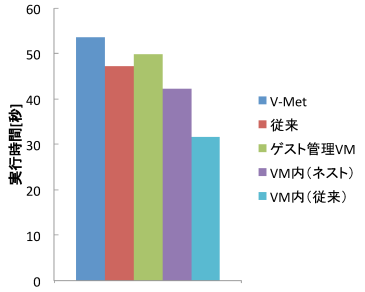

図 17　Tripwire の実行時間（ローカル）

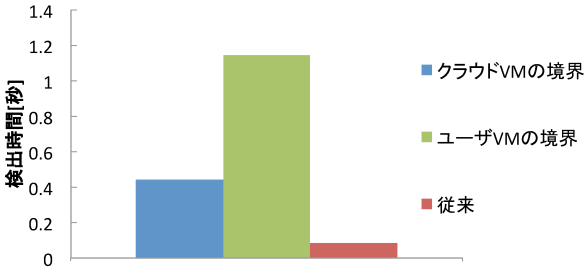

図 18　ポートスキャン検出時間

ポートスキャンを行い，その検出にかかる時間を測定した．検出時間は，ユーザ VM に対してポートスキャンを開始してから，Snort がポートスキャンを検出するまでの時間とした．実験結果は図 18 のようになり，クラウド VM の境界でパケットを取得した場合，従来システムより検出時間が 0.36 秒増加した．これは ebtables 経由で取得したパケットを V-Met がユーザ VM ごとに振り分けて tap デバイスに書き込むオーバヘッドである．従来システムでは仮想 NIC から直接パケットを取得できていた．一方，ユーザ VM の境界でパケットを取得した場合，検出時間は 1.06 秒増加した．これはユーザ VM が受信したパケットをクラウドハイパーバイザ経由でクラウド管理 VM に送ることによるオーバヘッドである．

5.4　オーバヘッド

V-Met においてユーザ VM が CR3 レジスタへの書き込みを行った際に，VM Exit を発生させることによりその値を保存するオーバヘッドを調べた．比較として，CR3 レジスタへのアクセスで VM Exit を発生させないデフォルト

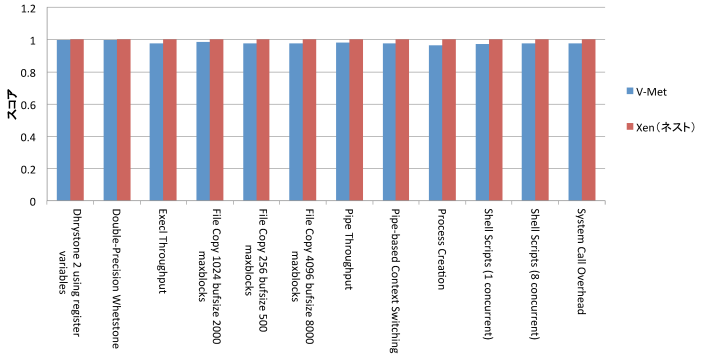

図 19　ユーザ VM における UnixBench のスコア

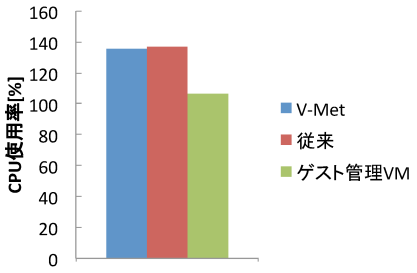

図 20　CPU 使用率（chkrootkit 実行時）

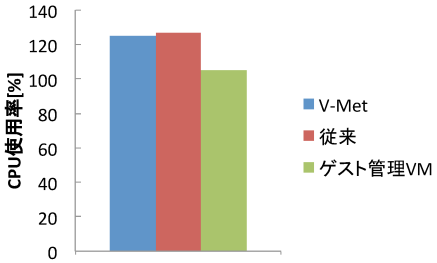

図 21　CPU 使用率（Tripwire 実行時）

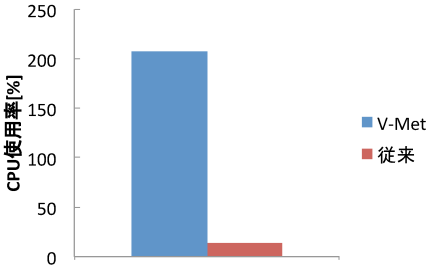

図 22　CPU 使用率（クラウド VM の境界で Snort 実行時）

設定でネストした仮想化を用いる Xen でも性能を測定した．図 19 にユーザ VM 内で UnixBench を実行した時の結果を示す．この結果は Xen でのスコアを 1 としたものである．VM Exit を発生させた場合には UnixBench のスコアが平均 2%低下することが分かった．

次に，IDS 実行中の CPU 使用率を測定した．図 20 と図 21 に示すように，chkrtootkit および Tripwire を実行している間，V-Met のクラウド管理 VM における CPU 使用率

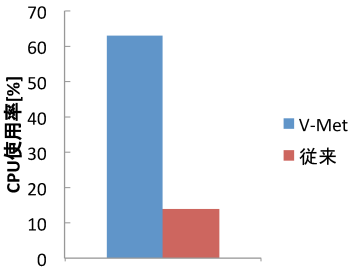

図 23　CPU 使用率（ユーザ VM の境界面で Snort 実行時）

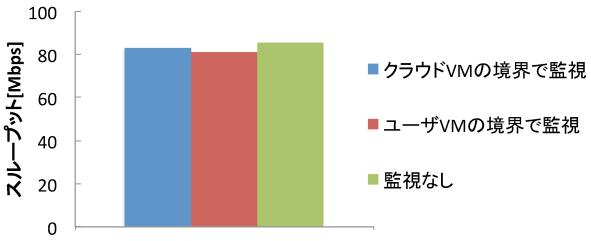

図 24　ネットワーク性能への影響

は従来システムの管理 VM より 1%だけ低くなった．ゲスト管理 VM では 28%および 21%低下した．Snort で監視しながら iperf を実行した時の CPU 使用率を図 22 と図 23 に示す．クラウド VM の境界で監視した場合，V-Met における CPU 使用率は従来システムの 15 倍となった．それに対して，ユーザ VM の境界で監視した場合は 4.5 倍となった．

また，V-Met におけるネットワーク監視がユーザ VM のネットワーク性能に及ぼす影響について調べた．クラウド VM の境界またはユーザ VM の境界で Snort を用いて監視を行った場合と監視を行わなかった場合とについて，ユーザ VM に対する iperf の性能を測定した．実験結果は図 24 のようになった．クラウド VM の境界で監視を行った場合，ユーザ VM のネットワーク性能は 3%低下した．一方，ユーザ VM の境界で監視を行った場合は性能が 6%低下した．

6. 関連研究

Self-Service Cloud (SSC) [3] 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．また，ハイパーバイザを信頼する必要がある．

RemoteTrans [6] はユーザ 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

で安全にユーザ 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 は信頼できるハイパーバイザと暗号通信を行い，ユーザ 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 [11] 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 [14] 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 [1] 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 [10] は Intel TXT や AMD SVM を用いて安全に IDS を実行するが，SMM と同様の問題を抱えている．

CloudVisor [15] 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，VM を安全に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 は従来の仮想化システム全体をクラウド VM 内で動作させ，クラウド VM 内のユーザ VM のメモリ，ディスク，ネットワークの監視を可能にする．V-Met を用いることで，IDS が仮想化システムの管理者に攻撃されるのを防ぐことができる．また，信頼できない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．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 のオーバヘッドが従来の IDS オフロードと同等であることを確認した．

今後の課題は，ユーザ VM の MAC アドレスや仮想ディスクを一意に特定できるようにすることである．現在の実装では，これらはあらかじめ分かっていることを仮定している．また，仮想化システム内のハイパーバイザや管理 VM など，ユーザ VM 以外も監視できるようにすることを計画している．

参考文献

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

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

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

[4] Garfinkel, T. and Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, *Proceedings of Network and Distributed Systems SecuritySymposium*, pp. 191–206 (2003).

[5] IntelCorp: 4th Generation Intel Core vPro Processors with Intel VMCS Shadowing (2013).

[6] Kourai, K. and Juda, K.: Secure Offloading of Legacy IDSes Using Remote VM Introspection in Semi-trusted Clouds, *Proceedings of the 9th IEEE International Conference on Cloud Computing*, pp. 43–50 (2016).

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

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

[9] LOWELL, D. E., SAITO, Y. and SAMBERG, E. J.: Devirtualizable virtual machines enabling general, single-node, online maintenance, *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 211–223 (2004).

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

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

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

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

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

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

[16] 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).