強制パススルー機構を用いたVMの安全な帯域外リモート管理

二神 翔太[1]　鵜木 智矢[1]　光来 健一[1]

概要：IaaS 型クラウドではユーザが仮想マシン（VM）にアクセスできるようにするために，帯域外リモート管理と呼ばれる機能を提供している．帯域外リモート管理は，VM の仮想デバイスを経由して VM に間接的にアクセスする管理手法である．仮想デバイスは仮想化システム内で動作しているが，クラウドにおいては仮想化システムを管理する管理者は必ずしも信頼できるとは限らない．悪意のある管理者は仮想デバイスから帯域外リモート管理の入出力を容易に盗聴することができる．従来，仮想化システム内のハイパーバイザを信頼することで管理者への情報漏洩を防ぐ手法が提案されてきたが，セキュリティや管理などの面で問題があった．そこで本稿では，**強制パススルー**と呼ぶ手法を用いて仮想化システムの外側で安全に帯域外リモート管理を実現する *VSBypass* を提案する．VSBypass ではネストした仮想化を用いて仮想化システム全体を VM 内で動作させ，ユーザの VM が行う入出力を横取りして仮想化システム外部の**シャドウデバイス**で処理する．これにより，帯域外リモート管理の入出力が仮想化システム内の管理者に漏洩することを防ぐ．我々は VSBypass を Xen に実装し，情報漏洩が防止できることの確認および，仮想シリアルコンソールを用いた帯域外リモート管理の性能測定を行った．

1. はじめに

Iaa 型クラウドでは，ユーザは必要な数の仮想マシン（VM）を使用し，大規模なシステムを構築することができる．クラウドによって提供された VM（ユーザ VM）を管理するために，ユーザは SSH や VNC などのリモート管理ソフトウェアを用いて遠隔地からアクセスを行う．クラウドでは，ユーザがネットワーク経由で直接 VM にアクセスする管理手法に加えて，帯域外リモート管理と呼ばれる管理手法を提供している．帯域外リモート管理は，仮想シリアルデバイスや仮想キーボードといった仮想デバイスを経由してユーザ VM にアクセスする手法である．この管理手法には，ユーザ VM のネットワーク障害時でも VM 内のシステム管理を行えるという利点がある．

しかし，仮想デバイスはクラウドのシステム管理者によって管理されているため，悪意あるシステム管理者は帯域外リモート管理の入出力を盗聴することができる．従来，ハイパーバイザを信頼してリモート管理クライアントとハイパーバイザの間で入出力を暗号化する手法が提案されてきた [1][2]．この手法により仮想デバイスからの情報漏洩を防ぐことができるが，仮想化システム内で管理者がハイパーバイザを攻撃するのは比較的容易であるという問題がある．その上，ハイパーバイザを信頼するためには，信頼できる管理者だけが仮想化システム全体を管理する必要がある．また，ハイパーバイザやリモート管理クライアントのサポートが必要なため，適用範囲が限定される．

そこで本稿では，**強制パススルー**と呼ぶ手法を用いて，仮想化システムの外側で安全に帯域外リモート管理を実現する *VSBypass* を提案する．VSBypass では，ネストした仮想化を用いて従来のクラウド環境における仮想化システム全体を VM 内で動作させる．ユーザ VM で入出力アクセスが行われると，仮想化システムの外側で入出力を横取りし，**シャドウデバイス**を用いて入出力の処理を行う．VSBypass では仮想化システム内のハイパーバイザを信頼する必要がなく，仮想化システム内の信頼できない管理者が帯域外リモート管理の入出力を盗聴することもできない．また，仮想化システムとリモート管理ソフトウェアには既存のものを利用することができる．

我々は仮想シリアルデバイスと仮想キーボード，仮想マウス，仮想ビデオカードの強制パススルーを行う VSBypass を Xen 4.8.0 に実装した．VM 内で動作させる仮想化システムとして Xen と KVM に対応している．仮想化システム内のユーザ VM が入出力アクセスを行うと，仮想化システムの外側で動作しているハイパーバイザに直接 VM Exit が発生する．この入出力を処理するシャドウデバイス

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

© 2017 Information Processing Society of Japan

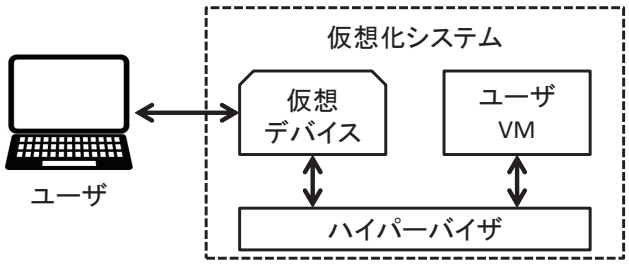

図 1 ユーザ VM の帯域外リモート管理

はユーザ VM と 1 対 1 に対応するように作成したプロキシ VM によって提供される．シャドウデバイスにおいて発生した仮想割り込みは仮想化システム経由でユーザ VM に転送される．実験により，VSBypass を用いることで仮想化システム内での盗聴を防ぐことができることを確認した．また，仮想化システムに Xen を用いた場合，仮想シリアルコンソールの応答時間が従来のクラウド環境より 1.3 ミリ秒長くなり，スループットは 2%低下することがわかった．

以下，2 章で帯域外リモート管理における情報漏洩と従来のアプローチの問題点について議論する．3 章でそれらの問題点をふまえて提案システムである VSBypass について述べる．4 章で VSBypass の実装について述べ，5 章で VSBypass と従来のクラウド環境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. 帯域外リモート管理における情報漏洩

帯域外リモート管理とは，図 1 のように，仮想化システム内の仮想デバイス経由で間接的にユーザ VM にアクセスする管理手法である．ユーザは SSH や VNC などのリモート管理クライアントを用いてユーザ VM の仮想デバイスにアクセスし，入出力情報のやりとりを行う．帯域外リモート管理に用いられる仮想デバイスとしては，仮想シリアルデバイスや仮想キーボード，仮想マウス，仮想ビデオカードがある．ユーザによって仮想デバイスに書き込まれた入力情報はハイパーバイザ経由でユーザ VM に渡される．一方，ユーザの出力情報はハイパーバイザ経由で仮想デバイスに書き込まれる．

帯域外リモート管理で用いられる仮想デバイスはクラウドのシステム管理者によって管理されている．しかし，クラウド事業者は信頼できるとしても，仮想化システムを管理しているシステム管理者は必ずしも信頼できるとは限らない．実際，サイバー犯罪の 28%は内部犯行であるという報告 [3] がある．信頼できない管理者としては悪意のあるシステム管理者が考えられ，例えば，Google の管理者がユーザのプライバシを侵害するという事件が発生している [4]．また，管理者の 35%は機密情報に無断でアクセスしたことがある [5] という報告からも分かるように，勤勉だが好奇心の強い管理者が存在することも知られている．こ

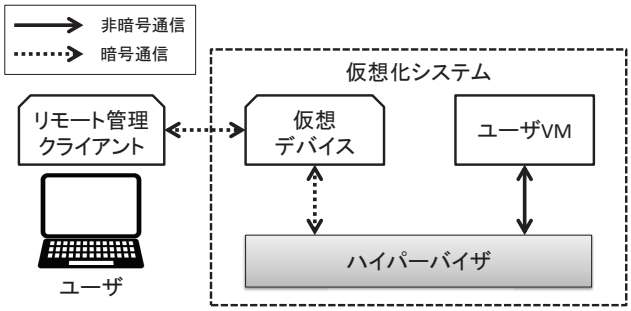

図 2 仮想デバイスに対する入出力の暗号化

のようなシステム管理者は仮想デバイスにおいて帯域外リモート管理の入出力情報を容易に盗聴することができる．その結果，ユーザ VM のログインパスワードや閲覧した文書などの機密情報が漏洩する恐れがある．

従来，リモート管理クライアントとハイパーバイザの間で入出力を暗号化することで，仮想デバイスにおける情報漏洩を防ぐ手法が提案されてきた．例えば，FBCrypt [1] や SCCrypt [2] は，図 2 のように，ユーザ VM の帯域外リモート管理を行う際に，SSH や VNC のクライアントにおいて入力情報を暗号化する．そして，ユーザ VM が仮想デバイスから入力情報を取得する際にハイパーバイザが復号化を行う．一方，ユーザ VM が出力情報を仮想デバイスに書き込む際にはハイパーバイザが暗号化を行い，それを受け取ったクライアントが復号化を行う．これにより，仮想化システム内で動作する仮想デバイスは暗号化データのみを扱うことになるため，入出力情報を盗聴することはできない．これらの従来の手法では，ハイパーバイザが信頼できることを前提としていた．

しかし，従来手法には以下の 4 つの問題点がある．

- **仮想化システム内でハイパーバイザを攻撃するのは比較的容易**
  仮想化システムはハイパーバイザ，管理コンポーネント，VM で構成されている．VM の管理ツールや仮想デバイスなどの管理コンポーネントをハイパーバイザ上で動作させるために，ハイパーバイザは様々な管理用 API を提供している．そのため，管理コンポーネントはハイパーバイザと密接に結びついており，ハイパーバイザへの攻撃を比較的行いやすい．
- **仮想化システムの管理が困難**
  ハイパーバイザを信頼できるようにするには，信頼できない管理者がハイパーバイザの管理を行わないことを保証する必要がある．しかし，確実に信頼できる管理者の人数は限られるため，信頼できる管理者だけが仮想化システム全体を管理すると負担が大きくなる．そこで，仮想化システムのハイパーバイザ以外の部分は信頼できない可能性がある管理者に管理させることが考えられる．その場合，仮想化システム全体の一括アップデートを行うことができなくなるため，仮想化

© 2017 Information Processing Society of Japan

システムの整合性を保つのが難しくなる．

- **特定の仮想化システムにのみ適用可能**
  ハイパーバイザだけを信頼するには，ハイパーバイザとそれ以外のソフトウェアが明確に分離されている必要がある．このような仮想化システムの例としては，ハイパーバイザ型の Xen や Hyper-V などが挙げられる．一方，ホスト型の KVM などではハイパーバイザがホスト OS 内で動作するため，ハイパーバイザとホスト OS を分離するのは難しい．ホスト OS 全体を信頼する方法も考えられるが，ホスト OS はハイパーバイザよりもはるかに複雑であるため，TCB が大きくなるという問題がある．
- **リモート管理クライアントへの変更が必要**
  仮想デバイスで処理される入出力情報を暗号化するには，リモート管理クライアントに入力の暗号化および出力の復号化の処理を追加する必要がある．そのため，商用クライアントなどの既存のクライアントが使えないという問題がある．

3. VSBypass

本稿では，**強制パススルー**と呼ぶ手法を用いて仮想化システムの外側で安全に帯域外リモート管理を実現する VSBypass を提案する．VSBypass では，ネストした仮想化を用いて仮想化システム全体を VM 内で動作させる．そして，ユーザ VM の入出力を横取りし，仮想化システム外部に用意した**シャドウデバイス**を用いて入出力処理を行う．これにより，VSBypass では従来の仮想化システムに依存せずに帯域外リモート管理を行うことができる．そのため，帯域外リモート管理の入出力情報が仮想化システム内の管理者に漏洩することはない．

VSBypass のシステム構成は図 3 のようになる．ユーザ VM が動いている従来の仮想化システムは**クラウド** *VM* と呼ばれる VM 内で動作させる．クラウド VM の外側では，強制パススルーによってユーザ VM がアクセスするシャドウデバイスを動作させる．シャドウデバイスは仮想化システム内の仮想デバイスと同様にユーザ VM の入出力処理を行う．クラウド VM やシャドウデバイスは**クラウドハイパーバイザ**上で動作させる．区別のために，仮想化システム内のハイパーバイザは**ゲストハイパーバイザ**と呼ぶ．

VSBypass における帯域外リモート管理の流れは以下のようになる．まず，ユーザ VM の仮想デバイスに対する入出力アクセスをクラウドハイパーバイザが直接，横取りする．そして，仮想化システムの外側で動作している強制パススルー先のシャドウデバイスを用いて入出力の処理を行う．シャドウデバイスで発生した仮想割り込みはユーザ VM に転送する．ユーザはリモート管理クライアントを用いてシャドウデバイスにアクセスすることにより，従来通りの帯域外リモート管理を行うことができる．

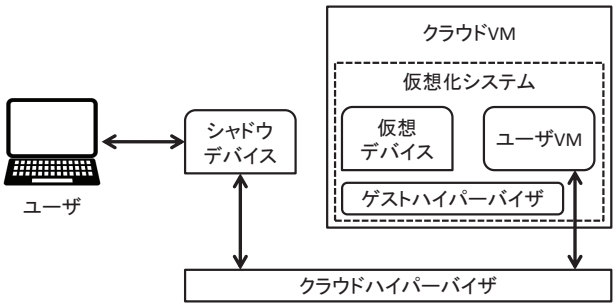

図 3 VSBypass のシステム構成

ネストした仮想化を用いることにより仮想化システムの性能が低下するが，オーバヘッドを削減する様々な手法が提案されている．それにより，Turtles Project [6] では一般的なワークロードの性能低下を 6%から 8%に抑えることができている．TinyChecker [7] では，軽量なハイパーバイザを用いることにより性能低下を約 1%に抑えている．Xen-Blanket [8] は，仮想ネットワークを準仮想化することにより，高速化する手法を提案している．VMCS Shadowing [9] のような CPU サポートを用いることでも，オーバヘッドを軽減することができる．

VSBypass では，信頼できない可能性のあるシステム管理者が仮想化システム全体を管理することを想定している．一方，仮想化システムの外側のシャドウデバイスやクラウドハイパーバイザはクラウド事業者が管理することを想定している．クラウド事業者が信頼できるという仮定は多くの研究 [10][11][12][13][14][15][16] で用いられており，一般的である．本稿では，信頼できないシステム管理者が仮想デバイスやゲストハイパーバイザにおいて帯域外リモート管理の入出力を盗聴したり改ざんしたりする攻撃を考える．ユーザ VM に対する攻撃は CloudVisor [12] で防ぐことを想定している．

VSBypass は従来の安全な帯域外リモート管理における様々な問題を解決することができる．第一に，VSBypass では仮想化システム内の信頼できない管理者がクラウド VM の外側のクラウドハイパーバイザやシャドウデバイスを攻撃するのは困難である．なぜなら，仮想化システム全体が VM 内で動作しており，VM とハイパーバイザ間のインタフェースは管理コンポーネントとハイパーバイザ間に比べて狭いためである．その分だけ，管理者が攻撃を行える可能性は低くなる．第二に，VSBypass では仮想化システム内の管理者が仮想化システム全体を管理することができる．仮想化システム内のハイパーバイザを信頼する必要がないため，信頼できない可能性がある管理者にもハイパーバイザを管理させることができる．仮想化システムの外側のシステムだけを信頼できるクラウド事業者が管理すればよいため，仮想化の境界できれいに管理を分担することができる．

第三に，VSBypass では仮想化システム全体を仮想化す

© 2017 Information Processing Society of Japan

るため，どのような仮想化システムであっても利用することができる．ハイパーバイザ型の仮想化システムだけでなく，ホスト型の仮想化システムであっても容易にサポートすることができる．これは仮想化システムにほぼ依存せずに帯域外リモート管理を実現するためである．第四に，VSBypass では帯域外リモート管理の入出力を暗号化しないため，ユーザは既存のリモート管理ソフトウェアを用いることができる．リモート管理クライアントを暗号化に対応させる必要はない．

4. 実装

我々は VSBypass を Xen 4.8.0 に実装した．VSBypass を実装したハイパーバイザをクラウドハイパーバイザとして動作させ，クラウド VM 内では既存のハイパーバイザをそのままゲストハイパーバイザとして動作させる．現在の実装では，VSBypass は仮想シリアルデバイス，仮想キーボード，仮想マウス，仮想ビデオカードの強制パススルーに対応しているが，仮想ビデオカードについては完全には動作していない．

4.1 プロキシ VM

VSBypass では図 4 のように，ユーザ VM ごとにプロキシ VM と呼ばれる VM をクラウドハイパーバイザ上で動作させる．プロキシ VM は，ユーザ VM に強制パススルー先の仮想デバイスを提供するためだけに用いられる VM である．プロキシ VM にはクラウド VM と同じ数の仮想 CPU を割り当てる．ユーザ VM が入出力を行う仮想 CPU はクラウド VM のいずれかの仮想 CPU に割り当てられるため，クラウド VM の仮想 CPU と 1 対 1 に対応づけることでプロキシ VM の仮想 CPU に入出力アクセスをリダイレクトしやすくするためである．一方，メモリやディスクの割り当ては最低限とし，NIC は割り当てない．ユーザはプロキシ VM の ID を指定してその仮想デバイス（シャドウデバイス）にアクセスすることで，対応するユーザ VM の帯域外リモート管理を透過的に行うことができる．

ユーザ VM とプロキシ VM を 1 対 1 に対応づけるために, ユーザ VM ごとに一つずつ作られる拡張ページテーブル（EPT）のアドレスを用いてユーザ VM を識別する．EPT のアドレスは入出力の横取り時にユーザ VM の VMCS から取得し，それが初めて検出された EPT のアドレスであれば新しいプロキシ VM と対応づける．現在の実装では，プロキシ VM をあらかじめ作成しておき，新しい EPT のアドレスが検出された時点でその内の一つを割り当てている．この時点で動的にプロキシ VM を作成できるようにするのは今後の課題である．

4.2 ユーザ VM の入出力の横取り

ネストした仮想化における従来の入出力の流れを図 5 に示す．ユーザ VM において入出力を行う命令を実行した時，そのユーザ VM が動作しているクラウド VM からクラウドハイパーバイザに対して VM Exit が発生する．クラウドハイパーバイザはクラウド VM への VM Entry を行い，ゲストハイパーバイザにおいて命令エミュレーションを行う．ゲストハイパーバイザは仮想デバイスと通信し，入出力の処理を行う．その後，ゲストハイパーバイザがユーザ VM に対して VM Entry を行うと，クラウドハイパーバイザへの VM Exit が発生し，クラウドハイパーバイザからユーザ VM に VM Entry を行う．

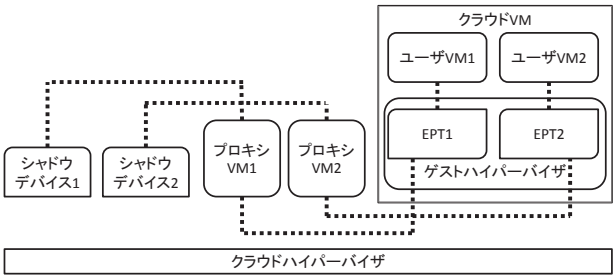

図 4 プロキシ VM を用いたシャドウデバイスの提供

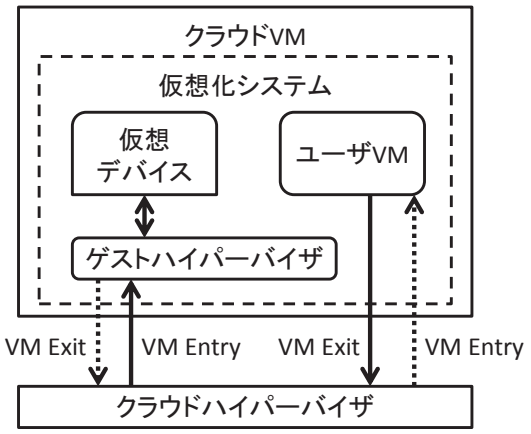

図 5 従来のネストした仮想化における入出力処理

VSBypass では，ユーザ VM の入出力アクセスをゲストハイパーバイザに転送せず，図 6 のようにクラウド VM の外側で処理する．ユーザ VM において入出力を行う命令を実行した時，従来と同様にクラウドハイパーバイザに対して VM Exit が発生する．入出力アクセスが I/O マップト I/O であった場合，クラウドハイパーバイザは VM Exit の原因となった入出力命令の対象ポート番号を調べる．それがシリアルデバイス，キーボード，マウス，ビデオカードによって使われているポート番号であった場合にはクラウドハイパーバイザにおいて命令エミュレーションを行う．一方，メモリマップト I/O であった場合には，EPT Violation の原因となった物理アドレスを調べる．それがビデオカードによって使われているアドレスであればクラウドハイパーバイザにおいて命令エミュレーションを行う．

クラウドハイパーバイザはユーザ VM に 1 対 1 に対応するプロキシ VM の仮想デバイス（シャドウデバイス）に入

© 2017 Information Processing Society of Japan

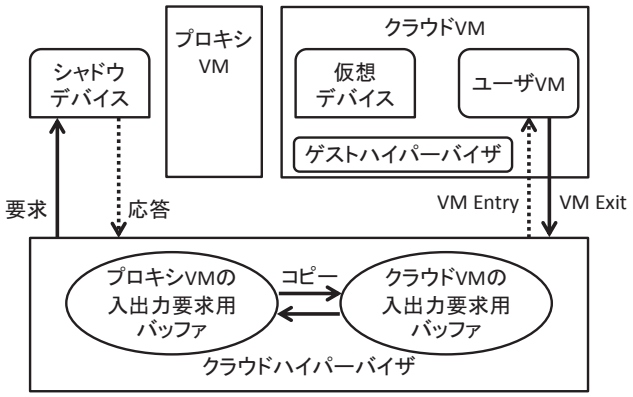

図 6 VSBypass における入出力処理

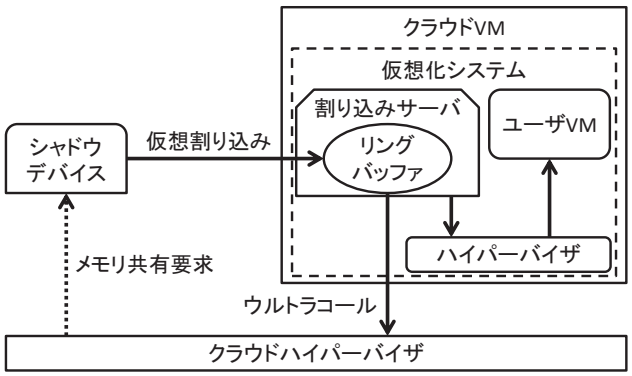

図 7 仮想割り込みの転送

出力要求を送るために，ユーザ VM の VMCS から EPT のアドレスを取得し，それを基に対応するプロキシ VM を見つける．次に，入出力アクセスの際に使われていたクラウド VM の仮想 CPU と同じ CPU 番号を持つプロキシ VM の仮想 CPU を見つける．そして，クラウド VM の仮想 CPU がもつ入出力要求用バッファの内容をプロキシ VM の仮想 CPU がもつバッファにコピーする．最後に，プロキシ VM の仮想 CPU をブロック状態にし，シャドウデバイスにイベントを送信する．

VSBypass では入出力処理の結果を以下のようにしてユーザ VM に返す．シャドウデバイスは入出力処理を完了すると，クラウドハイパーバイザ経由でプロキシ VM にイベントを送信する．クラウドハイパーバイザはイベントの送信先がプロキシ VM であり，処理した入出力要求がシャドウデバイスに対するものであった場合，対応するユーザ VM が動作しているクラウド VM を見つける．次に，プロキシ VM の仮想 CPU がもつ入出力要求用バッファから，対応するクラウド VM の仮想 CPU がもつバッファに入出力の実行結果と状態をコピーする．最後に，その仮想 CPU のブロック状態を解除し，ユーザ VM への VM Entry を行って入出力アクセスの次の命令からユーザ VM の実行を再開する．

4.3 ユーザ VM への仮想割り込みの転送

シャドウデバイスで発生した仮想割り込みはユーザ VM へ転送しなければならないが，仮想化システムに依存せずに行うのは難しい．割り込み機構が準仮想化されている場合はゲストハイパーバイザしか仮想割り込みを送ることができず，準仮想化されていない場合でも割り込みコントローラがゲストハイパーバイザ内に実装されていることが多いためである．できるだけ仮想化システムに依存しないようにするには，シャドウデバイスと仮想化システムの間でネットワーク通信を行い，ゲストハイパーバイザ経由で仮想割り込みを転送する方法が考えられる．しかし，この方法は仮想ネットワークのオーバヘッドが大きい．Xen-Blanket [8] では Blanket ドライバを用いてこの通信を高速化しているが，ドライバを組み込み，ハイパーコールを追加する必要があるため，仮想化システムに大きく依存する．

そこで VSBypass では，図 7 のようにウルトラコール [16] と呼ばれる機構を用いてシャドウデバイスと仮想化システムの間でリングバッファを共有し，ポーリングに基づく通信を行う．仮想化システム内で動作する割り込みサーバはウルトラコールを用いて直接クラウドハイパーバイザに制御を移すため，ゲストハイパーバイザへの変更は不要である．そして，ウルトラコールで渡されたメモリ情報をシャドウデバイスに転送することで，リングバッファを共有する．リングバッファの読み書きを通知できるようにするには仮想化システムの改変が必要となるため，リングバッファを定期的にポーリングすることで割り込み情報の取得を行う．これにより，Xen を用いた仮想化システムでは割り込みサーバのプロセスを実行するだけで済む．KVM では実装上の問題で QEMU に割り込みサーバを組み込む必要があるが，OS やハイパーバイザを改変する必要はない．

シャドウデバイスで発生した仮想割り込みをユーザ VM に転送する処理の流れを図 7 に示す．シャドウデバイスは発生した仮想割り込みの IRQ 番号と電圧レベルをリングバッファに書き込む．割り込みサーバはリングバッファからこれらの情報を読み出し，ハイパーコールを発行してゲストハイパーバイザ経由で仮想割り込みをユーザ VM に送る．このように，仮想割り込みの転送は信頼できない割り込みサーバおよびゲストハイパーバイザに依存しているため，正しくユーザ VM に送られない可能性がある．しかし，仮想割り込みには機密情報は含まれていないため，情報が漏洩する恐れはない．

5. 実験

VSBypass において，帯域外リモート管理の入力が盗聴できないことを確認する実験を行った．また，帯域外リモート管理における入力の応答時間と出力のスループットを測定した．比較対象として，ネストした仮想化を用いない従来のクラウド環境，ネストした仮想化を用いるが強制パス

© 2017 Information Processing Society of Japan

スルーは行わない従来の仮想クラウド環境，仮想割り込みの転送に仮想ネットワークを使う VSBypass を用いた．

仮想化システムに Xen を用いた実験では，Intel Xeon E3-1290v2 の CPU，8GB のメモリ，1TB の HDD を搭載したマシンを用いた．クラウド VM 内では Xen 4.4.0 を動作させ，仮想化システム内の管理 VM とユーザ VM では Linux 3.13 を動作させた．クラウド VM には 2 個の仮想 CPU と 4GB のメモリ，ユーザ VM には 2 個の仮想 CPU と 1GB のメモリをそれぞれ割り当てた．仮想化システムに KVM を用いた実験では，Intel Xeon E3-1226v3 の CPU，8GB のメモリ，500GB の HDD を搭載したマシンを用いた．クラウド VM 内では QEMU-KVM 2.4.1 を動作させ，クラウド VM とユーザ VM の OS として Linux 4.2 を用いた．クラウド VM には 2 個の仮想 CPU と 3GB のメモリ，ユーザ VM には 2 個の仮想 CPU と 1GB のメモリをそれぞれ割り当てた．どちらの実験でもクラウドハイパーバイザとして VSBypass または Xen 4.8.0 を動作させた．一方，リモート管理クライアントを動作させるために，Intel Xeon E3-1270 の CPU，8GB のメモリ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イーサネットで接続した．

5.1 仮想デバイスにおける入出力情報の盗聴

VSBypass および従来のクラウド環境において帯域外リモート管理の入力が仮想化システム内の仮想デバイスで盗聴できるかどうかを確認する実験を行った．仮想シリアルコンソールを用いる場合は，SSH クライアントでサーバに接続し，シャドウデバイスまたは仮想シリアルデバイスにアクセスした．GUI リモートアクセスを用いる場合には，VNC クライアントでサーバに接続し，シャドウデバイスまたは仮想キーボードにアクセスした．いずれの場合も，従来のクラウド環境ではユーザが入力した文字を仮想デバイスにおいて盗聴することができた．一方，VSBypass では仮想デバイスにおいては盗聴することができなかった．

5.2 応答時間

帯域外リモート管理における入力に対する応答時間を測定した．応答時間は SSH クライアントで文字を入力してからその文字がユーザ VM によって処理され，エコーバックされた文字が SSH クライアントに表示されるまでの時間とした．仮想化システムとして Xen と KVM を用いた時の測定結果をそれぞれ図 8, 図 9 に示す．

仮想化システムに Xen を用いた場合，VSBypass では従来のクラウド環境と比べて，応答時間が 1.3 ミリ秒長くなった．ネストした仮想化のオーバヘッドによりユーザ VM 内での処理に時間がかかったことが原因と考えられる．仮想割り込みの転送に仮想ネットワークを用いた場合は，そのオーバヘッドのために応答時間が 3.3 ミリ秒長くなった．

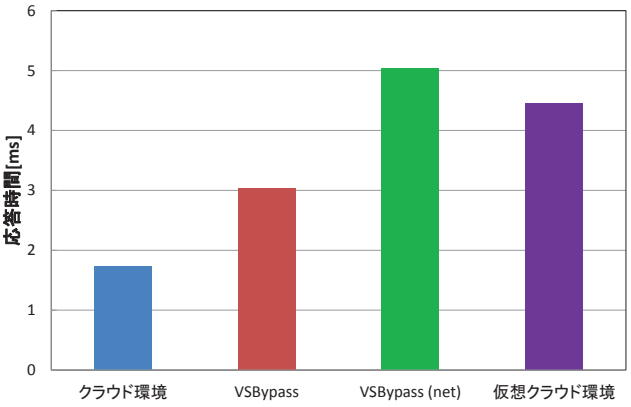

図 8 Xen における応答時間

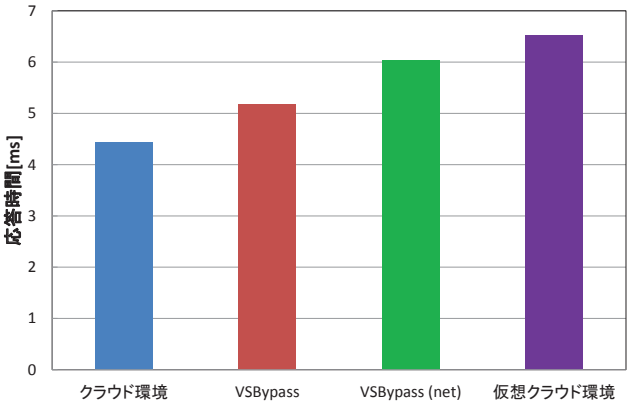

図 9 KVM における応答時間

一方，従来の仮想クラウド環境と比較すると応答時間が 1.4 ミリ秒短くなった．これは VSBypass ではシャドウデバイスにおいてネストした仮想化のオーバヘッドを被らなかったためだと考えられる．仮想化システムに KVM を用いた場合も Xen を用いた場合とほぼ同様の結果となったが，全体的に Xen よりも応答時間が長くなり，性能差は小さくなった．仮想割り込みの転送に仮想ネットワークを用いた場合の性能は Xen とは異なり，仮想クラウド環境よりもよくなった．これは KVM における仮想ネットワーク性能が Xen よりも高いためと考えられる．

5.3 スループット

帯域外リモート管理を用いてログインしたユーザ VM で cat コマンドを実行し，テキストファイルの中身を表示した時のスループットを測定した．コマンドを実行してからテキストファイルを表示し終わるまでの時間を SSH クライアントにおいて測定し，スループットを算出した．仮想化システムに Xen と KVM を用いた場合の測定結果をそれぞれ図 10，図 11 に示す．

仮想化システムに Xen を用いた場合，VSBypass では従来のクラウド環境と比べて，スループットの低下は 2%だけであった．仮想割り込みの転送に仮想ネットワークを用いると 52%低下したため，リングバッファにより大幅に高

© 2017 Information Processing Society of Japan

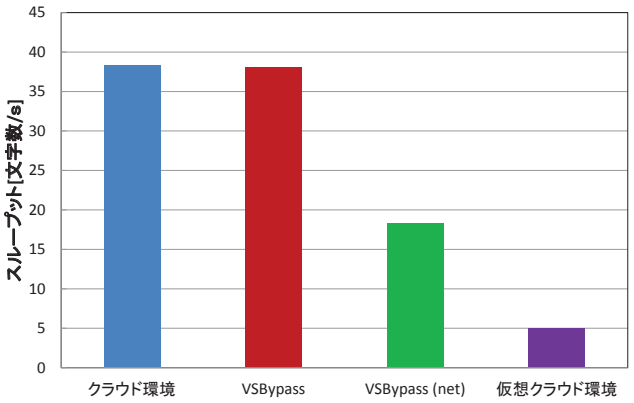

図 10 Xen におけるスループット

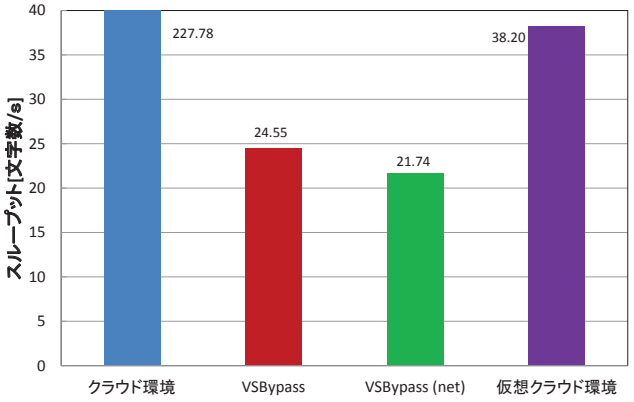

図 11 KVM におけるスループット

速化できていることが分かる．一方，従来の仮想クラウド環境と比較すると，スループットは 7.5 倍に向上した．この実験は仮想デバイスに大きな負荷をかけるため，仮想デバイスがネストした仮想化のオーバヘッドを被る場合には性能が著しく低下したと考えられる．仮想化システムに KVM を用いた場合，従来のクラウド環境でのスループットが非常に高く，VSBypass ではスループットが 89%低下した．仮想クラウド環境でのスループットも VSBypass より高かったため，VSBypass の実装上の問題だと考えられる．また，仮想割り込みの転送に仮想ネットワークを用いた場合と比べると，リングバッファを用いても 11%しか性能が向上しなかった．

6. 関連研究

デバイスパススルーは VM から物理デバイスに直接アクセスすることを可能にする機構である．デバイスを仮想化することによるオーバヘッドを削減し，VM の入出力性能を向上させるために用いられる．PCI，VGA，GPU，HDD，NIC など，様々なデバイスに対してパススルーを用いることができる．VSBypass で用いる強制パススルーは二つの点で従来のパススルーとは異なる．第一に，VM にパススルーの設定を行うことなく，強制的にパススルーが行われる．第二に，パススルーによってアクセスされるデバイスが物理デバイスではなく，仮想デバイスである．そのため，すべての VM に対して同じ種類の仮想デバイスにパススルーでアクセスさせることができる．

BitVisor [17] は準パススルーと呼ばれる機構を提供し，必要に応じて VM によるパススルーでのデバイスアクセスをハイパーバイザが横取りすることができる．この機構を用いて，BitVisor は暗号化などのセキュリティ機能を実現したり，透過的なネットワークブートを実現したりしている．VSBypass ではパススルーを行う VM を動作させているゲストハイパーバイザで横取りをするのではなく，その下のクラウドハイパーバイザで横取りする点が異なる．

CloudVisor [12] はネストした仮想化を用いて仮想化システムの下にセキュリティモニタを導入することで，ユーザ VM を仮想化システム内の信頼できない管理者から守る．VSBypass と同様に，信頼できない管理者が仮想化システム内の仮想デバイスおよびハイパーバイザを管理することを想定している．CloudVisor では，管理者がユーザ VM のメモリにアクセスするのを制限することができる．また，メモリやディスクを暗号化したり，ハッシュ値を付加したりすることで，情報漏洩や改ざんを防ぐことができる．しかし，帯域外リモート管理で用いられる仮想シリアルデバイスや仮想キーボード，仮想ビデオカードについては保護を行っていない．

VMware vSphere [18] はハイパーバイザ内で仮想デバイスと VNC サーバを動作させるため，ハイパーバイザが信頼できる場合は仮想デバイスからの情報漏洩を防ぐことができる．しかし，信頼できない管理者がハイパーバイザを管理している場合には，入出力情報が漏洩する恐れがある．VSBypass では，従来の仮想化システム内のハイパーバイザが入出力を扱わないようにすることで，ハイパーバイザにおける情報漏洩を防ぐことができる．

FBCrypt [1] と SCCrypt [2] は，帯域外リモート管理を行う際に入出力情報の漏洩を防ぐ．VNC や SSH などのリモート管理クライアントにおいて入力を暗号化し，仮想デバイスからユーザ VM に入力が渡される際にハイパーバイザが復号する．FBCrypt では入力の改ざんを検出するために，メッセージ認証コードを用いて整合性の検査も行う．一方，ユーザ VM の出力は仮想デバイスに渡される際にハイパーバイザによって暗号化され，リモート管理クライアントによって復号される．VSBypass では仮想化システム内の仮想デバイスで入出力処理を行わないため，このような暗号化や整合性検査は不要である．FBCrypt や SCCrypt では準仮想化された入出力デバイスにも対応しているが，このような仮想デバイスにアクセスしても VM Exit が発生しないため，VSBypass では対応するのが難しい．

© 2017 Information Processing Society of Japan

7. まとめ

本稿では，強制パススルーと呼ぶ手法を用いて，仮想化システムの外側で安全に帯域外リモート管理を実現するVSBypass を提案した．VSBypass では，ネストした仮想化を用いて仮想化システム全体を VM 内で動作させる．そして，ユーザ VM の入出力を仮想化システムの外側で横取りし，シャドウデバイスで処理する．これにより，仮想化システム内の管理者は帯域外リモート管理における入出力情報を盗聴することができない．仮想シリアルデバイス，仮想キーボード，仮想マウス，仮想ビデオカードについて強制パススルーを実装し，帯域外リモート管理の性能を測定した．

今後の課題は，VNC を用いた帯域外リモート管理に対応することである．現在のところ，仮想キーボードの強制パススルーは正常に動作している．仮想ビデオカードについては入出力を横取りすることはできているが VNC クライアントに表示することができていない．

参考文献

[1] T. Egawa, N. Nishimura, and K. Kourai: Dependable and Secure Remote Management in IaaS Clouds, *Proc. Int. Conf. Cloud Computing Technology and Science*, pp. 411–418 (2012).
[2] K. Kourai and T. Kajiwara: Secure Out-of-band Remote Management Using Encrypted Virtual Serial Consoles in IaaS Clouds, *Proc. Int. Conf. Trust, Security and Privacy in Computing and Communications*, pp. 443–450 (2015).
[3] PwC: US Cybercrime: Rising Risks, Reduced Readiness (2014).
[4] TechSpot News: Google Fired Employees for Breaching User Privacy, http://www.techspot.com/news/40280-google-fired-employees-for-breaching-user-privacy.html (2010).
[5] CyberArk Software: Global IT Security Service (2009).
[6] M. Ben-Yehuda and M. D. Day and Z. Dubitzky and M. Factor and N. Har'El and A. Gordon and A. Liguori and O. Wasserman and B.-A. Yassour: The Turtles Project: Design and Implementation of Nested Virtualization, *Pro. Operating Systems Design and Implementation*, pp. 423–436 (2010).
[7] C. Tan and Y. Xia and H. Chen and B. Zang: TinyChecker: Transparent Protection of VMs against Hypervisor Failures with Nested Virtualization, *Pro. In. Workshop on Dependability of Clouds, Data Centers and Virtual Machine Technology* (2012).
[8] D. Williams and H. Jamjoom and H. Weatherspoon: The Xen-Blanket: Virtualize Once, Run Everywhere, in Proc. European Conf. Computer Systems, pp.113-126 (2012).
[9] Intel Corp.: 4th Generation Intel Core vPro Processors with Intel VMCS Shadowing (2013).
[10] N. Santos and K. P. Gummadi and R. Rodrigues: Towards Trusted Cloud Computing, *Proc. Workshop on Hot Topics in Cloud Computing* (2009).
[11] C. Li and A. Raghunathan and N. K. Jha: Secure Virtual Machine Execution under an Untrusted Management OS, *Proc. Int. Conf. Cloud Computing*, pp. 172–179 (2010).
[12] Zhang, F., Chen, J., Chen, H. and Zang, B.: CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization, *Proc. Symp. Operating Systems Principles*, pp. 203–216 (2011).
[13] Tadokoro, H., Kourai, K. and Chiba, S.: Preventing Information Leakage from Virtual Machines' Memory in IaaS Clouds, *IPSJ Online Trans.*, Vol. 5, pp. 156–166 (2012).
[14] Li, C., Raghunathan, A. and Jha, N. K.: A Trusted Virtual Machine in an Untrusted Management Environment, *IEEE Trans. Services Computing*, Vol. 5, No. 4, pp. 472–483 (2012).
[15] Butt, S., Lagar-Cavilla, H. A., Srivastava, A. and Ganapathy, V.: Self-service Cloud Computing, *Proc. Conf. Computer and Communications Security*, pp. 253–264 (2012).
[16] Miyama, S. and Kourai, K.: Secure IDS Offloading with Nested Virtualization and Deep VM Introspection, *Proc. European Symp. Research in Computer Security, part II*, pp. 305–323 (2017).
[17] T. Shinagawa and S. Hasegawa and T. Horie and Y. Oyama and S. Chiba and H. Eiraku and K. Tanimoto and M. Hirano and E. Kawai and Y. Shinjo and K. Omote and K. Kourai and K. Kono and K. Kato: BitVisor: A Thin Hypervisor for Enforcing I/O Device Security, Proc. Int. Conf. Virtual Execution Environments, pp.121-130 (2009).
[18] VMware Inc: VMware vSphere, http://www.vmware.com/.

© 2017 Information Processing Society of Japan