ネストした仮想化を用いたVMの安全な帯域外リモート管理

二神 翔太[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，管理 VM と呼ばれる VM を経由してユーザの VM に間接的にアクセスする管理手法である．管理 VM はクラウドのシステム管理者によって管理されているが，クラウド事業者は信頼できるとしても，そのシステム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管理 VM 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．そこで本研究では，ネストした仮想化と呼ばれる技術を用いて，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，従来のクラウド環境で用いられていた仮想化システムを VM 内で動作させ，その中のユーザ VM で行われる入出力を**強制パススルー**と呼ばれる機構を用いて仮想化システムの外側で処理する．これにより，帯域外リモート管理の入出力情報がクラウドのシステム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，SSH 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，管理 VM と呼ばれる VM を経由して間接的にユーザ 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，帯域外リモート管理で経由する管理 VM 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ユーザ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し，管理 VM への情報漏洩を防ぐ手法が提案されてきた [1][2]．しかし，この手法にはいくつかの問題がある．第一に，仮想化システム全体を一人のシステム管理者が管理できなくなる．第二に，管理 VM からハイパーバイザを攻撃するのは比較的容易である．第三に，特定の仮想化システムのみに適用可能である．第四に，リモート管理クライアントへの変更が必要である．

そこで本稿では，ネストした仮想化と呼ばれる技術を用いて，従来のクラウド環境における仮想化システムを VM 内で動作させ，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，ユーザ VM で入出力命令が実行されると，仮想化システムの外側で入出力命令を横取りして，専用の仮想デバイスを用いて入出力の処理を行う．これを**強制パススルー**と呼ぶ．入出力はこの仮想デバイスとリモート管理クライアントの間でやり取りされる．そのため，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はできない．

我々は仮想シリアルコンソールをサポートした VSBypass を Xen 4.4.0 に実装した．ホスト VM 内で動作するユーザ VM が入出力命令を実行すると，ホストハイパーバイザへの VM Exit が発生する．ホストハイパーバイザはその入出力命令を横取りし，ホスト管理 VM 上のにある強制パススルー先の仮想シリアルデバイスで入出力処理を行う．この仮想シリアルデバイスはユーザ VM と 1 対 1 に対応するように作成したプロキシ VM が提供する．仮想シリアルデバイスにおいて発生した仮想割り込みはホスト VM 内の仮想化システムに転送することでユーザ VM に送る．実験を行い，VSBypass を用いることで従来の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，SSH を用いた帯域外リモート管理の応答時間が従来のクラウド環境より約 3.5 ミリ秒長くなり，スループット

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

© 2012 Information Processing Society of Japan

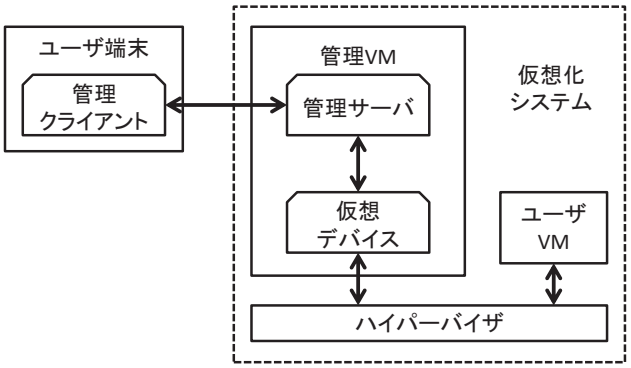

図 1 ユーザ VM の帯域外リモート管理

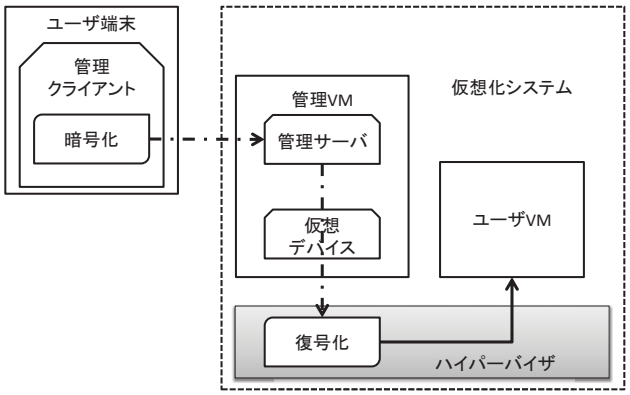

図 2 SCCrypt のシステム構成

は約 12%に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 章では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 章では 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 章では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管理 VM 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管理 VM 内のリモート管理サーバにアクセスし，ユーザ VM の仮想デバイスとの間で入出力情報のやりとりを行う．管理 VM においてエミュレートされてい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，仮想ビデオカードなどがある．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，出力情報はユーザ VM から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経由する管理 VM 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，管理 VM 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] もある．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．このようなシステム管理者は管理 VM 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．それにより，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帯域外リモート管理の入出力を暗号化することで，管理 VM における情報漏洩を防ぐ手法が提案されてきた．例えば，SCCrypt [2] は，図 2 のように，SSH を用いてユーザ VM の帯域外リモート管理を行う際に，SSH 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 が仮想シリアル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 が出力情報を仮想シリアルデバイスに書き込む際にハイパーバイザが暗号化を行い，それを受け取った SSH クライアントが復号化を行う．これにより，管理 VM で動作する仮想シリアルデバイスやリモート管理サーバは暗号化データのみを扱うことになるため，入出力情報を盗むことはできない．FBCrypt [1] は VNC の入出力について同様の暗号化を行う．これらの従来の手法では，前提としてハイパーバイザを信頼していた．

しかし，従来手法における問題点として，以下の 4 点が挙げられる．

- **仮想化システム全体を管理 VM のシステム管理者に管理させられない**
  管理 VM を信頼しないがハイパーバイザは信頼する場合，管理 VM とハイパーバイザを別々の管理者が管理することになる．そのため，例えば，管理 VM の管理者は仮想化システム全体のアップデートを行うことができない．これはデバイスエミュレータや管理ツールのみをアップデートすることができ，ハイパーバイザのアップデートができないためである．しかし，仮想化システムは通常，一つのソフトウェアパッケージとして提供されており，仮想化システム全体を同時にアップデートできないと整合性の問題が生じる．一方で，信頼できる管理者だけが仮想化システム全体のアップデートを行えるようにすると負担が大きくなる．
- **管理 VM からハイパーバイザを攻撃するのは比較的容易**
  管理 VM はハイパーバイザが提供する様々な API を

© 2012 Information Processing Society of Japan

用いて VM の管理を行ったり，仮想デバイスを動作させたりするため，ハイパーバイザと密接に結びついている．そのため，通常の VM からに比べて，管理 VM からはハイパーバイザへの攻撃を比較的行いやすくなっている．

- **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管理コンポーネント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**
  管理 VM 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，既存のクライアントや商用クライアントが使えないという問題がある．また，リモート管理クライアントが標準的にサポートしない限りは，バージョンアップへの対応を行い続ける必要がある．

3. VSBypass

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帯域外リモート管理を実現する VSBypass を提案する．ネストした仮想化は，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．VSBypass は，ユーザ VM の入出力を横取りし，仮想化システム外部の仮想デバイスを用いて入出力処理を行う．これを**強制パススルー**と呼ぶ．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．従来の仮想化システムはホスト VM と呼ばれる VM 内で動作させる．ホスト VM を管理するための VM はホスト管理 VM と呼ばれ，ホスト管理 VM 内ではユーザ VM に強制パススルーでアクセスさせる仮想デバイスおよび帯域外リモート管理に用いるリモート管理サーバを動作させる．これらのホスト VM はホストハイパーバイザ上で動作させる．ホスト VM 内で動作する従来の仮想化システム内の管理 VM とハイパーバイザは，区別のために，それぞれゲスト管理 VM，ゲストハイパーバイザと呼ぶ．

ネストした仮想化は従来の仮想化システムの性能を低下させるが，オーバヘッドを削減する様々な手法が提案されている．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，軽量なホスト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．また，VMCS Shadowing [9] のような CPU サポートを用いることでも，オーバーヘッドを軽減することができる．

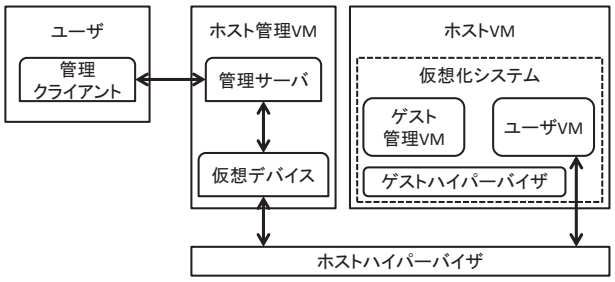

図 3 VSBypass のシステム構成

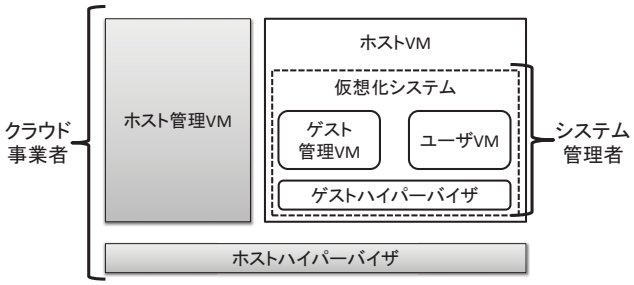

図 4 管理の分担

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 の仮想デバイスへの入出力命令をホストハイパーバイザが直接，横取りする．そして，ホスト管理 VM で動作させる強制パススルー先の仮想デバイスを用いて入出力の処理を行う．ユーザはリモート管理ソフトウェアを用いてこの仮想デバイスにアクセスすることにより，帯域外リモート管理を行う．一方，この仮想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．

VSBypass では，図 4 に示すように信頼できないシステム管理者がゲスト管理 VM とゲストハイパーバイザを管理し，信頼できるクラウド事業者がホスト管理 VM とホストハイパーバイザを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仮定は多くの研究 [10][11][12][13][14][15] 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ゲスト管理 VM 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 では管理 VM のシステム管理者が仮想化システム全体を管

© 2012 Information Processing Society of Japan

理できる．管理 VM のシステム管理者がハイパーバイザも管理できるため，従来と同じ管理方法を用いることができる．その一方で，仮想化システムの外側のシステムは別の信頼できる管理者だけが管理することができる．このように，仮想化の境界できれいに分担して管理を行うことができる．第二に，VSBypass では仮想化システム内の信頼できない管理者がホスト VM の外側を攻撃して帯域外リモート管理の入出力情報を盗むのは困難である．なぜなら，仮想化システム全体が通常の VM 内で動作しており，通常の VM とハイパーバイザ間のインタフェースは管理 VM とハイパーバイザ間に比べて狭いためである．それだけ，管理者が攻撃を行える可能性は低くなる．

第三に，VSBypass では従来の仮想化システム全体を仮想化するため，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特定の仮想化システム以外であっても，ホスト型の仮想化システムでさえもサポートすることができる．これは仮想化システムに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 4.4.0 に実装した．VSBypass 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．現在の実装では，VSBypass は SSH 経由で仮想シリアルコンソールにアクセスする帯域外リモート管理に対応している．

4.1 プロキシ VM

VSBypass では図 5 のように，ユーザ VM ごとにプロキシ VM と呼ばれる VM をホスト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シリアルデバイスを提供するだけのために用いられるホスト VM である．ユーザ VM の各仮想 CPU で実行される入出力命令をそれぞれ中継するために，プロキシ VM にはユーザ VM と同じ数の仮想 CPU を割り当てる．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 を指定してホスト管理 VM 上の仮想シリアル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命令の横取り時にユーザ VM の

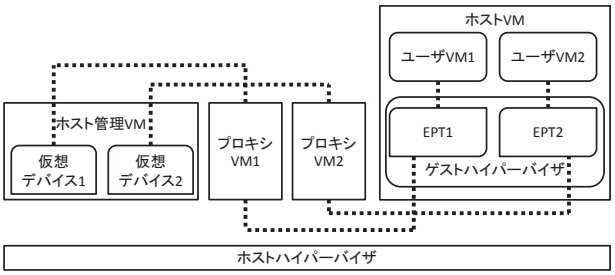

図 5 プロキシ VM を用いたユーザ VM への仮想デバイスの提供

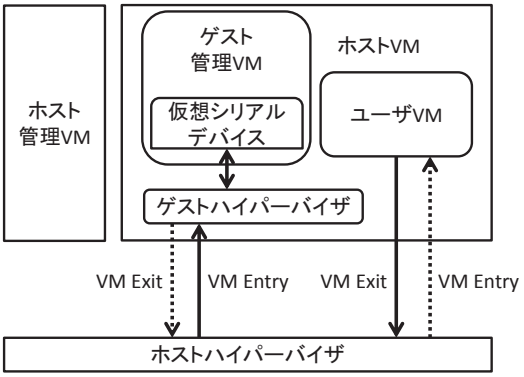

図 6 従来のネストした仮想化における入出力処理

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 の入出力命令の横取り

ネストした仮想化における従来の入出力の流れを図 6 に示す．ユーザ VM において入出力命令を実行した時，そのユーザ VM が動作しているホスト VM からホストハイパーバイザに対して VM Exit が発生する．ホストハイパーバイザはホスト VM への VM Entry を行い，ゲストハイパーバイザにおいて入出力命令のエミュレーションを行う．ゲストハイパーバイザはゲスト管理 VM 内で動作している強制パススルー先の仮想シリアルデバイスと通信し，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，ホストハイパーバイザへの VM Exit が発生し，ホストハイパーバイザからユーザ VM に VM Entry を行う．

VSBypass では図 7 の実線のように，ユーザ VM の入出力命令をゲストハイパーバイザに転送せず，ホスト VM の外側で処理する．ユーザ VM において入出力命令を実行した時，従来と同様にホストハイパーバイザに対して VM Exit が発生する．ホストハイパーバイザは VM Exit の原因となった入出力アクセスの対象ポート番号を調べ，シリアルデバイスのポート番号であった場合にはホストハイ

© 2012 Information Processing Society of Japan

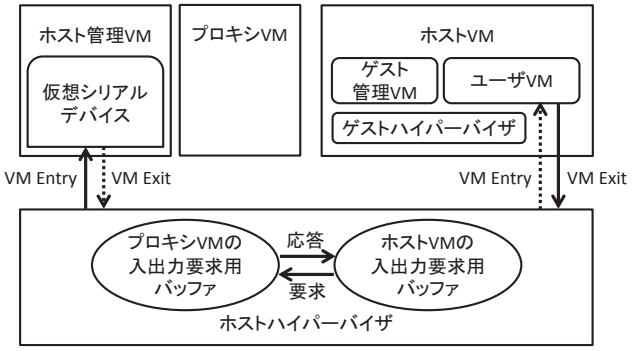

図 7 VSBypass における入出力処理

パーバイザにおいて入出力処理を行う．ホスト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 の仮想シリアルデバイスに入出力要求を送るために，まず，ユーザ VM の VMCS から EPT のアドレスを取得し，対応するプロキシ VM を見つける．次に，入出力命令を実行したユーザ VM の仮想 CPU に対応するプロキシ VM の仮想 CPU を見つけ，仮想 CPU とに割り当てられた入出力要求用バッファに入出力要求を書き込む．最後に，プロキシ VM の仮想 CPU をブロック状態にし，対応する仮想シリアルデバイスにイベントを送信する．

VSBypass では図 7 の点線のように，入出力処理の結果を以下のようにしてユーザ VM に返す．ホスト管理 VM 上の仮想シリアルデバイスは入出力処理を完了すると，ホストハイパーバイザにイベントを送信する．ホスト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リアルデバイスに対するものであった場合，そのプロキシ VM に対応するユーザ VM が動作しているホスト VM を見つける．次に，プロキシ VM の仮想 CPU に割り当てられた入出力要求用バッファから，そのホスト VM の対応する仮想 CPU に割り当てられた入出力要求用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 を行って入出力命令の次の命令からユーザ VM の実行を再開する．

4.3 仮想割り込みのユーザ VM への転送

強制パススルー先の仮想シリアル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 8 に示す．まず，仮想シリアル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ネットワーク通信によりゲスト管理 VM で動作させた割り込みサーバに送る．割り込みサーバはハイパーコールを発行し，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．ゲストハイパーバイザ経由で仮想割り込みを送るのは，ユーザ VM のゲスト OS が準仮想化の割り込み機構を用いているためである．このように，仮想割り込みの転送は信頼できないゲスト管理 VM およびゲストハイパーバイザに依存しているため，

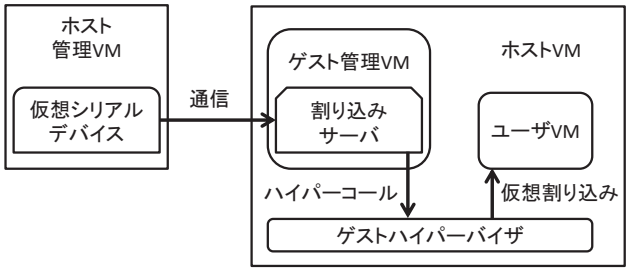

図 8 仮想割り込みの転送

```
(XEN) (HOST)VMX_OUT:a
(XEN) (HOST)VMX_OUT:b
(XEN) (HOST)VMX_OUT:c
(XEN) (HOST)VMX_OUT:d
(XEN) (HOST)VMX_OUT:e
(XEN) (HOST)VMX_OUT:f
(XEN) (HOST)VMX_OUT:g
```

図 9 従来のクラウド環境での盗聴結果

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．完全仮想化の割り込み機構を用いたゲスト OS に対応するのは今後の課題である．

5. 実験

VSBypass において，SSH 経由で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入出力を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既存のクラウド環境，および，従来のネストした仮想化を用いた仮想クラウド環境を用いた．

実験には，CPU に Intel Xeon E3-1290v2，メモリを 8GB，HDD を 1TB 搭載したマシンを用いた．ホストハイパーバイザには VSBypass を実装した Xen 4.4.0 を動作させ，ホスト VM 内では既存の Xen 4.4.0 を動作させた．ホスト管理 VM およびゲスト管理 VM，ユーザ VM では Linux 3.13 を動作させた．一方，SSH クライアントを動作させるために，CPU に Intel Xeon E3-1270，メモリを 8GB 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 ハイパーバイザによ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出力を盗聴する実験を行った．VSBypass においてはゲストハイパーバイザで盗聴を行い，従来のクラウド環境においてはハイパーバイザで盗聴を行った．図 9 に示すように，従来のクラウド環境ではユーザが入力した文字をハイパーバイザにおいて盗聴することができた．一方，VSBypass ではゲストハイパーバイザにおいて盗聴することはできなかった．

© 2012 Information Processing Society of Japan

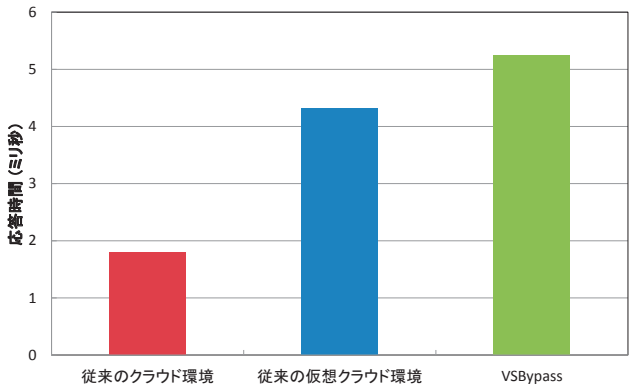

図 10 応答時間

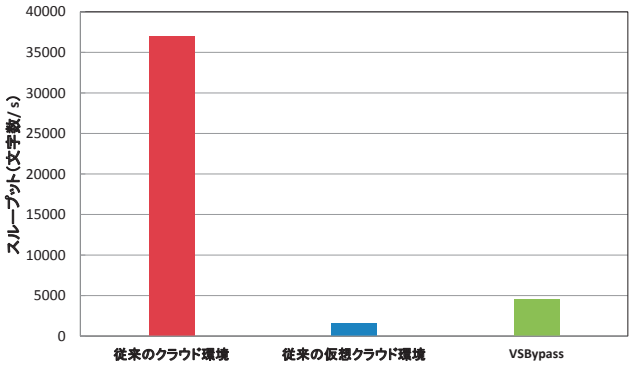

図 11 スループット

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 クライアントにおいて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，SSH クライアントに表示されるまでの時間とした．図 10 に測定結果を示す．VSBypass では従来のクラウド環境と比べて，応答時間が約 3.5 ミリ秒長くなった．ネストした仮想化のオーバーヘッドおよび仮想割り込みの転送時のネットワーク通信が原因と考えられる．一方，従来の仮想クラウド環境と比較しても，応答時間が約 1 ミリ秒長くなった．このオーバヘッドは仮想割り込みの転送時のネットワーク通信によるものであると考えられる．

5.3 スループット

帯域外リモート管理を用いてゲスト 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．図 11 に測定結果を示す．VSBypass では従来のクラウド環境と比べて，スループットが 12%にまで低下した．この原因も，応答時間が長くなった原因と同じであると考えられるが，システムにより高い負荷がかかったために性能低下がより顕著になったと思われる．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 2.7 倍に向上した．

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ことである．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ことである．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 [16] 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 では強制パススルーによって仮想デバイスにアクセスするため，容易にデバイスアクセスを横取りして付加的な処理を行うことができる．

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管理 VM 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 [17] はハイパーバイザ内で VNC サーバを動かすため，管理 VM を経由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，管理 VM の管理者へ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] および SCCrypt [2] は，IaaS 型クラウド内の VM に対して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システムである．VNC や SSH などのリ

© 2012 Information Processing Society of Japan

モート管理クライアントにおいて入力を暗号化し，管理VM内の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，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．

7. まとめ

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帯域外リモート管理を実現するVSBypassを提案した．VSBypassでは，従来の仮想化システム全体をVM内で動かしてユーザVMの入出力命令を横取りし，強制パススルーを行うためにホスト管理VM上の仮想デバイスで処理する．これにより，仮想化システムの管理者は従来通りハイパーバイザを管理することができるが，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はできない．

今後の課題としては，実験によって判明したオーバヘッドを改善することが挙げられる．例えば，ネットワーク通信を用いずに仮想割り込みを転送することが考えられる．また，SSHだけでなくVNCを用いた帯域外リモート管理にも対応する予定である．そのためには，仮想キーボードや仮想ビデオカードもホスト管理VMで動作させられるようにする必要がある．

参考文献

[1] Egawa, T., Kourai, K. and Nishimura, N.: Secure Out-of-band Remote Management in IaaS Clouds, *IPSJ Transactions on Advanced Computing Systems*, pp. 106–117 (2012).
[2] Kourai, K. and Kajiwara, T.: 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, *Proceedings of the 9th USENIX Symposium on 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, *Proceedings of the 2nd IEEE/IFIP International 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 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. Intl. 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] 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).
[17] VMware Inc: VMware vSphere, http://www.vmware.com/.

© 2012 Information Processing Society of Japan