クラウドにおけるVMリダイレクト攻撃を防ぐためのリモート管理機構

猪口 恵介　光来 健一

近年，IaaS 型クラウドの利用が広まってきており，ユーザは IaaS 型クラウドによって提供される仮想マシン（VM）に管理サーバ経由でアクセスする．しかし，管理サーバを管理するクラウド管理者は必ずしも信頼できるとは限らず，VM に対して様々な攻撃を行う恐れがある．考えられる攻撃の一つとして，ユーザのアクセス先 VM を変更する VM リダイレクト攻撃が考えられる．本稿では，このような VM リダイレクト攻撃を防止する UVBond を提案する．UVBond は VM の暗号化ディスクを利用することでユーザと VM を強く結びつけ，セキュアな VM 識別子を発行する．そして，ユーザによって指定された管理コマンドがその VM 識別子に対応する VM に対して実行されることを保証する．我々は UVBond を Xen に実装し，正しい VM 識別子を指定し，かつ，指定された管理コマンドを正しく実行した場合のみ VM を操作可能であることを確認した．

1　はじめに

近年，クラウドコンピューティングの利用が広まってきており，その利用形態の一つである IaaS 型クラウドでは，ユーザに仮想マシン（VM）の提供を行う．ユーザは VM の管理を行う場合，まずネットワークを介してクラウドの管理サーバに接続を行い，そこから VM へアクセスを行う．例えば，ユーザは管理サーバを通して VM の起動や終了，マイグレーションなどの操作，VNC や SSH などのリモート管理システムを用いた帯域外リモート管理などを行う．

管理サーバは VM の管理者とは異なるクラウドのシステム管理者によって管理されているが，クラウド管理者は必ずしも信頼できるとは限らない．そのため，信頼できないクラウド管理者が管理サーバの権限を悪用し，VM に対して様々な攻撃を行うことが考えられる．考えられる攻撃の一つとして，ユーザがアクセスする先の VM を意図的に変更し，クラウド管理者によって用意された悪意ある VM に接続させる攻撃が考えられる．本稿では，このような接続先の VM を変更させる攻撃を *VM* **リダイレクト攻撃**と呼ぶ．VM リダイレクト攻撃によって悪意ある管理者が用意した VM にアクセスさせられた場合，VM 内部にインストールされたマルウェアなどによってユーザの機密情報を盗まれる恐れがある．

本稿では，VM の暗号化ディスクを介してユーザと VM を強く結びつけることにより VM リダイレクト攻撃を防ぐ *UVBond* を提案する．UVBond はディスク暗号化を利用してユーザの VM を安全に起動し，VM が正しく起動されたことをユーザ自身が確認することを可能にする．VM が起動されると，UVBond はユーザにセキュアな VM 識別子の発行を行い，ユーザはこの VM 識別子と管理コマンドを指定してリモート管理を行う．その際に，UVBond は指定された管理コマンドが VM 識別子に対応する VM に対して実行されることを保証する．これらの処理を VM を動作させる基盤ソフトウェアであるハイパーバイザ内で実行することでその安全性を担保する．

我々は UVBond を Xen 4.4.0 [2] に実装した．UVBond ではハイパーバイザに VM の暗号化ディスクの暗号鍵を登録し，準仮想化ドライバを用いる OS に対してハイパーバイザ内部でディスクの暗号化・復号化を実行する．管理コマンドの識別には，管理コマンドがハイパーバイザに対して発行するハイ

Keisuke Inokuchi, Kenichi Kourai, 九州工業大学, Kyushu Institute of Technology.

パーコールの列を利用する．正しいハイパーコール列が実行されている間だけ，VM 識別子によって指定された VM へのアクセスを許可する．実験により，クラウド管理者がアクセス先の VM や実行する管理コマンドを意図的に変更することができないことを確認した．また，VM の起動時間とディスク I/O 性能を測定し，UVBond では VM の起動時間が 5.7 秒長くなり，読み込み性能が 9.5%，書き込み性能が 3.2%低下することが分かった．

以下，2 章でクラウドにおける管理サーバの信頼性と VM リダイレクト攻撃について述べ，3 章で提案するシステムである UVBond について述べる．4 章で実装の詳細について述べ，5 章で行った実験について述べる．6 章で関連研究に触れ，7 章でまとめと今後の課題について述べる．

2 VM リダイレクト攻撃

クラウドにおいて，ユーザはクラウド内の管理サーバを経由して VM を管理する．管理サーバは VM に対して起動や停止，バックアップの作成，他のホストへのマイグレーションなどの様々な操作を行うことができる．また，VM の仮想デバイスに直接アクセスすることにより，VM の帯域外リモート管理を行うこともできる．帯域外リモート管理は VM のネットワークに依存せず，SSH や VNC などを用いて VM 内のシステムにログインしてアクセスすることを可能にする．

管理サーバはクラウド管理者によって管理されるが，クラウド管理者は必ずしも信頼できるとは限らない．例えば，Google の管理者がユーザの機密情報を無断で閲覧し，ユーザのプライバシを侵害するという事件 [16] が発生している．また，サイバー犯罪の 28%は内部犯行であり [11]，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 [4] という報告もある．このように，クラウドプロバイダ自体は信頼することができるとしても，クラウド内にその管理権限を悪用する管理者が存在しないことを保証するのは難しい．

信頼できないクラウド管理者は管理サーバの権限を悪用し，ユーザの VM に対して様々な攻撃を行う可能性がある．例えば，クラウド管理者は VM のメモリ内の情報や帯域外リモート管理の入出力の情報を盗み見ることができる．このような情報漏洩を防ぐために，クラウド管理者に対して VM のメモリや入出力を暗号化する手法が提案されてきた [5] [7] [8] [9] [15]．これらの手法では，管理サーバや VM の下で動作するハイパーバイザと呼ばれるソフトウェアを信頼することにより，クラウド管理者に対して VM の情報を隠蔽する．

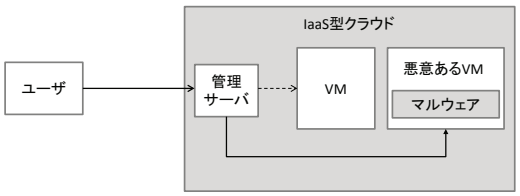

図 1　VM リダイレクト攻撃

本稿では，信頼できないクラウド管理者によって行われる新たな攻撃として VM リダイレクト攻撃を考える．VM リダイレクト攻撃は図 1 のように，管理サーバにおいてユーザがアクセスする VM を変更する攻撃である．クラウド管理者はマルウェアなどをインストールした悪意ある VM を作成しておき，ユーザをその VM にアクセスさせることによって様々な攻撃を行うことができる．例えば，システムのログインプログラムを改ざんしたり，キーロガーを仕掛けることによってユーザがシステムにログインする際のパスワードを盗むことができる．この攻撃はアクセス先の VM 内部で行われるため，VM の帯域外リモート管理における入出力を暗号化したとしても防ぐことはできない．

3 UVBond

3.1 脅威モデル

本稿では，クラウド内に信頼できない管理者が存在する可能性があり，クラウド全体が信頼できるとは限らない状況を想定する．信頼できない管理者は管理サーバの権限を悪用してクラウド内の VM に自由にアクセスすることができる．本稿では特に，従来手法では対処が難しかった VM リダイレクト攻撃を考

える．

一方で，仮想化システムの根幹となるハイパーバイザおよびその下のハードウェアは信頼できるものとする．このような仮定は様々な研究において行われている [3] [19] [13] [8] [15] [9]．信頼できるハイパーバイザは様々な手法によって実現可能である．例えば，TPM を用いたセキュアブートによってハイパーバイザが正常に起動したことを保証することができる．また，ハードウェアを用いた監視手法により，実行時の改ざんを検出することもできる [12] [10] [17] [1]．

3.2 ユーザと VM の結び付け

本稿では，VM の暗号化ディスクを介してユーザと VM を強く結びつけることで VM リダイレクト攻撃を防ぐ UVBond を提案する．UVBond は従来行われていた仮想デバイスレベルや VM 内でのディスク暗号化ではなく，クラウド管理者が干渉できないハイパーバイザレベルでのディスク暗号化を用いる．クラウドにおいて情報漏洩を防ぐためには従来でもディスクを暗号化する必要があったため，UVBond においてディスク暗号化を用いてもオーバヘッドが大きく増大することはない．

UVBond では，VM の起動時にユーザとハイパーバイザの間でディスク暗号鍵を共有し，ハイパーバイザ内で VM とディスク暗号鍵を結び付ける．この時，ディスク暗号鍵はハイパーバイザの公開鍵によって暗号化して送られるためクラウド管理者には漏洩せず，ハイパーバイザ内の秘密鍵でのみ復号できる．UVBond は登録されたディスク暗号鍵を用いて図 2 のようにハイパーバイザ内でディスクの暗号化・復号化を行いながら VM を起動する．登録されたディスク暗号鍵に対応する暗号化ディスク以外では VM を起動できないため，ユーザの VM が起動されることを保証することができる．

さらに，UVBond は登録されたディスク暗号鍵がユーザのものであるかの確認を行う．これは，クラウド管理者がディスク暗号鍵をハイパーバイザへの登録の際にすり替えることが可能なためである．信頼できない管理者が不正な暗号化ディスクを作成し，そのディスク暗号鍵を登録することで不正な VM を起動される危険性がある．ハイパーバイザは VM 起動の際に，図 3 のように確認用のデータを登録されたディスク暗号鍵で暗号化してユーザに返送する．このデータを受け取ったユーザが正しく復号できれば，登録されたディスク暗号鍵はユーザのものであることが確認できる．

VM が起動された時に，ハイパーバイザは VM に結び付けられたセキュアな VM 識別子をユーザに対して発行する．VM 識別子は登録されたディスク暗号鍵によって暗号化され，ユーザに返送される．VM 識別子は図 4 のようにユーザが VM にアクセスする

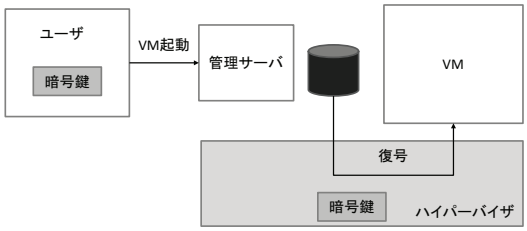

図 2　暗号化ディスクを用いた VM の起動

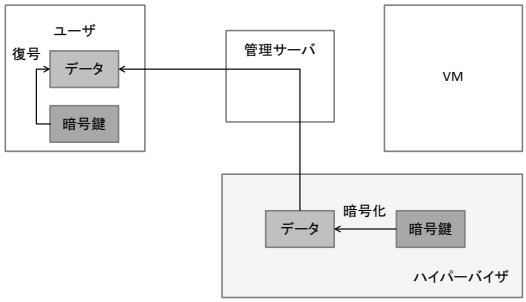

図 3　登録されたディスク暗号鍵の確認

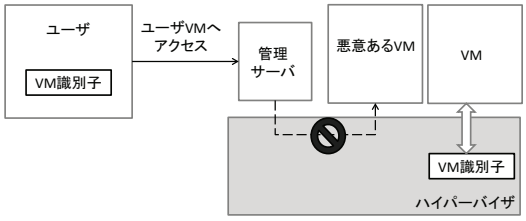

図 4　VM 識別子を用いたリモート管理

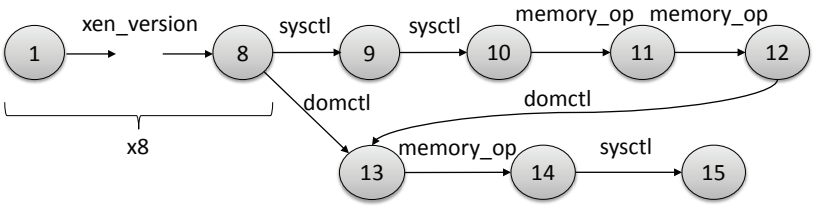

図 5　VM のメモリ設定コマンドのハイパーコール列

際に使用され，ユーザが指定した VM 以外にアクセスすることを防ぐ．そのため，信頼できない管理者が VM リダイレクト攻撃を行おうとしても，リダイレクト先の VM へのアクセスはハイパーバイザによって拒否される．

3.3　VM の安全なリモート管理

起動した VM を操作するために，ユーザは管理サーバ経由で管理コマンドを実行する．その際に VM 識別子を指定することで，UVBond は対応する VM に対して指定した管理コマンドが実行されることを保証する．しかし，VM 識別子を用いたアクセス制御はハイパーバイザが行う一方で，ハイパーバイザは管理コマンドを認識することができない．管理コマンドは管理サーバにおいて実行され，ハイパーバイザに対してハイパーコールを発行しながら VM の操作を行う．ハイパーバイザはハイパーコールしか認識できないため，管理コマンドの実行に直接 VM 識別子を結びつけることは難しい．

そこで UVBond では，ハイパーコールの呼び出しを入力とする図 5 のようなオートマトンを用いて管理コマンドを識別する．本稿ではこのオートマトンをハイパーコール列と呼ぶ．ユーザは VM 識別子とともに，指定した管理コマンドに対応するハイパーコール列を暗号化してハイパーバイザに送信する．ハイパーバイザは送られてきたハイパーコール列に沿って管理コマンドが実行されている間だけ，VM 識別子に対応する VM へのアクセスを許可する．それぞれの管理コマンドによって呼び出されるハイパーコールは必ずしも固定ではなく，状況によって変化する場合もある．例えば，VM のメモリサイズを設定する管理コマンドは初回と 2 回目以降の実行で呼び出されるハイパーコールが異なる．そのため，ハイパーコール列は管理コマンドによって呼び出される可能性があるすべてのハイパーコール呼び出しを含む．

ハイパーコールの呼び出し順が同じであったとしても，ユーザが意図した管理コマンドが実行されているとは限らない．信頼できないクラウド管理者はユーザがハイパーバイザに登録したハイパーコール列に受理されるようにハイパーコールを呼び出す別の管理コマンドを実行することが可能である．しかし，VM への操作は必ずハイパーコール経由で行われるため，ハイパーコールの呼び出し順が同じであれば VM に対して本質的に同じ操作しか行うことができない．そのため，ユーザが意図した管理コマンドとは異なるとしても，ハイパーコール列が同じであれば同じ管理コマンドとみなすことができる．ただし，登録されるハイパーコール列が複雑になればなるほど，元の管理コマンドとは異なる挙動をする管理コマンドの実行が容易になる．また，VM のメモリへのアクセスのように，ハイパーコールでアクセスできるようにした後，そのメモリへのアクセスをハイパーバイザがチェックできない場合には，ユーザが意図しない操作が行われる可能性がある．

UVBond はクラウド管理者にも VM への一部の操作を許可することができる．これは，クラウド管理者が VM をまったく管理できなくなると不便なためである．例えば，VM の終了やマイグレーションだけを許可することが考えられる．VM 識別子を持たないクラウド管理者が VM を操作できるようにするために，実行を許可する管理コマンドのハイパーコール列をあらかじめハイパーバイザに登録しておく．クラ

ウド管理者であっても，この管理コマンドが実行されている間は特定の VM への操作が許可される．許可する管理コマンドはユーザが自由に決定可能であり，管理者にどの程度の操作を許可するかによって管理とセキュリティのトレードオフをとることができる．

4 実装

我々は UVBond を Xen 4.4.0 [2] に実装した．Xen では管理サーバはドメイン 0 と呼ばれる VM 内で動作し，管理者もドメイン 0 を用いて VM の管理を行う．ユーザの VM はドメイン U と呼ばれる．ハイパーバイザ内での暗号化・復号化のために WolfSSL [18] の AES と RSA を移植し，ユーザ端末のクライアントでは OpenSSL を使用した．

4.1 VM のライフサイクル

UVBond を用いた場合の VM のライフサイクルは以下のようになる．ユーザは VM の作成時に AES 方式のディスク暗号鍵を生成し，この暗号鍵を用いて VM のディスクイメージを暗号化する．暗号化されたディスクイメージはクラウドにアップロードされ，従来通りドメイン 0 内に格納される．

VM を起動する際に，クライアントは VM の起動コマンドとともにハイパーバイザの RSA 公開鍵によって暗号化されたディスク暗号鍵をドメイン 0 に送信する．ハイパーバイザの公開鍵は信頼できる鍵サーバから取得するが，現在はあらかじめクライアントに登録している．暗号化されたディスク暗号鍵を受け取ったドメイン 0 はハイパーコールを発行して受信したデータをハイパーバイザに渡す．ハイパーバイザは自身の秘密鍵でこのデータを復号し，起動しようとしている VM にディスク暗号鍵を登録する．ディスクイメージの暗号化から暗号鍵の復号までのプロセスを図 6 に示す．

ドメイン 0 は起動コマンドに基づき，ユーザが指定した暗号化ディスクを用いて VM の起動を行う．VM がディスクにアクセスする際に，ハイパーバイザがディスクアクセスを横取りし，読み込もうとしているデータの復号化，書き込もうとしているデータの暗号化を行う．この処理の詳細については 4.2 節，4.3 節で述べる．VM の起動と並行して，クライアントはハイパーバイザに登録されたディスク暗号鍵の正しさを確認する．この詳細については 4.4 節で述べる．

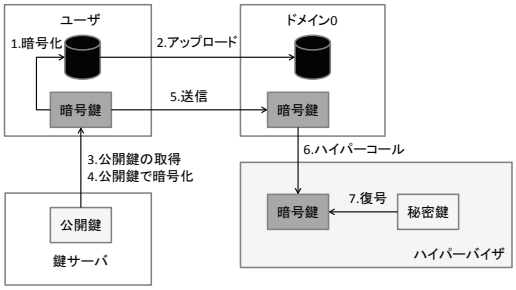

図 6　ディスクのアップロードと暗号鍵の登録

VM が起動すると，ドメイン 0 はハイパーコールを発行してハイパーバイザからセキュアな VM 識別子を取得する．この VM 識別子はハイパーバイザによって生成され，VM に登録されているディスク暗号鍵を用いて暗号化される．ドメイン 0 が VM 識別子をクライアントに送信すると，クライアントはディスク暗号鍵を用いてそれを復号する．クライアントが VM に対して管理コマンドを実行する際にはこの VM 識別子と管理コマンドのハイパーコール列の組を暗号化して送信する．この詳細については 4.5 節で述べる．

4.2 準仮想化ディスク I/O の暗号化

VM のディスクアクセスには準仮想化ディスクドライバが用いられることが多い．完全仮想化ディスクドライバを用いる場合と比べて，ディスク入出力の性能を向上させることができるためである．Xen は準仮想化と完全仮想化の両方のゲスト OS をサポートしているが，完全仮想化の場合でも Linux では準仮想化ディスクドライバが用いられる．

4.2.1 従来のディスク入出力

Xen の準仮想化ディスクドライバは，図 7 のようにドメイン U で動作する blkfront ドライバとドメイン 0 で動作する blkback ドライバからなる．これらのドライバは I/O リングと呼ばれるリングバッファが格納されたメモリ領域を共有し，イベントチャネル

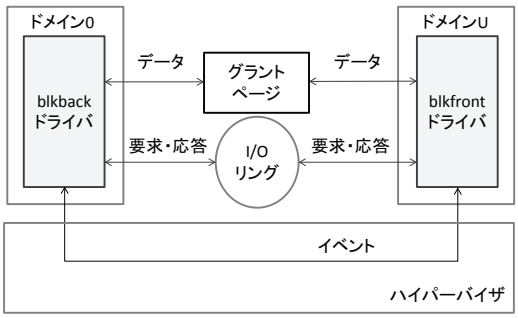

図 7　従来のディスク入出力

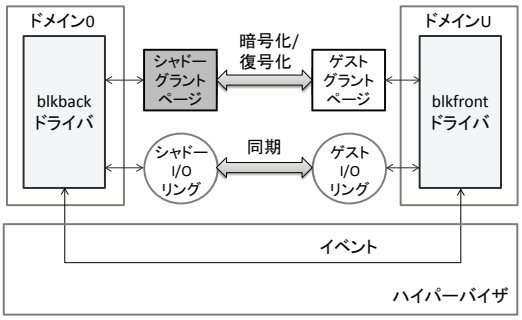

図 8　共有ページの二重化

と呼ばれる機構を用いて通信を行う．VM がディスクの入出力を行う際には，blkfront ドライバが I/O リングに要求を書き込み，イベントを送信する．イベントを受け取った blkback ドライバはその要求を I/O リングから取り出し，VM のディスクイメージにアクセスすることで入出力を実現する．読み込みの場合には，要求で指定されたドメイン U のバッファ領域にディスクイメージから読み込んだデータを書き込む．書き込みの場合は，指定されたバッファ領域のデータをディスクイメージに書き込む．入出力が完了すると，blkback ドライバは応答を I/O リングに書き込み，blkfront ドライバにイベントを送信する．

ドメイン 0 とドメイン U の間でのメモリ領域の共有には，グラントテーブルと呼ばれる機構が用いられる．ドメイン U はまず，グラントテーブルに共有したいメモリページを登録する．このページはグラントページと呼ばれる．グラントページにはグラント参照が割り当てられ，ドメイン 0 はグラント参照を指定して対応するページをマップしてアクセスする．I/O リングが格納されているページのグラント参照は XenStore と呼ばれるドメイン 0 内のデータベース経由で blkfront ドライバから blkback ドライバに渡される．ディスクの読み書きに用いられるバッファ領域が格納されているページについては，I/O リングに書き込まれる要求を通してグラント参照が渡される．

4.2.2　共有ページの二重化

ハイパーバイザが復号した後のデータをドメイン 0 に傍受されないようにするために，UVBond は図 8 のようにバッファ領域を格納しているグラントページを二重化する．ドメイン U には従来通り，暗号化されていないページを提供するが，ドメイン 0 には暗号化されたページを提供する．ドメイン U に提供するページをゲストグラントページと呼び，ドメイン 0 に提供するページをシャドーグラントページと呼ぶ．

一方，I/O リングを格納しているグラントページについても二重化は行うが，暗号化は行わない．この二重化は，ハイパーバイザが暗号処理を終えた要求だけを blkback ドライバに渡し，復号処理を終えた応答だけを blkfront ドライバに渡せるように同期をとるために行う．ドメイン U に提供する I/O リングをゲスト I/O リング，ドメイン 0 に提供する I/O リングをシャドー I/O リングと呼ぶ．

UVBond は従来との互換性を保つために，ドメイン U から渡されたグラント参照を用いて，二重化されたグラントページにアクセスすることを可能にする．ドメイン U に対しては従来通り，グラント参照をゲストグラントページに対応させる．一方，ドメイン 0 に対しては，同じグラント参照をシャドーグラントページに対応させる．これにより，ドメイン 0 がグラント参照に対してメモリマップ処理を行うと，ゲストグラントページの代わりにシャドーグラントページがマップされ，アンマップ処理を行うとシャドーグラントページがアンマップされる．ドメイン 0 からはゲストグラントページにアクセスできているように見えるが，実際にはシャドーグラントページにアクセスすることになる．

4.2.3 暗号化ディスクを用いた入出力

UVBond では，VM の起動時にハイパーバイザがドメイン U とドメイン 0 の間の通信を解析することによって I/O リングが格納されたメモリページを特定し，シャドー I/O リングを作成する．blkfront ドライバは初期化時に，XenStore に I/O リングが格納されたページのグラント参照を登録する．この際に，XenStore リングと呼ばれるリングバッファに要求を書き込んでイベントを送るため，イベントを送信するハイパーコール内で XenStore リングから I/O リングのグラント参照を取得する．XenStore リングが格納されているページの情報は VM の起動時にハイパーバイザに通知される．そして，シャドー I/O リングを作成した後で，ゲスト I/O リングの中身をシャドー I/O リングにコピーする．blkfront ドライバが用いるイベントチャネルのポート番号も同様にして取得する．

blkfront ドライバから blkback ドライバに要求が送られた時，それがディスクへの書き込み要求であればハイパーバイザはデータの暗号化を行う．要求を送る際にはイベントを送信するためのハイパーコールが呼ばれるため，その中でゲスト I/O リングに書き込まれた要求を解析する．要求に含まれるグラント参照に対応するシャドーグラントページがまだなければ作成する．ディスクへの書き込み要求の場合には，ゲストグラントページ上のバッファ領域に格納されたデータを暗号化しながらシャドーグラントページに書き込む．最後に，この要求をシャドー I/O リングにコピーする．

blkback ドライバから blkfront ドライバに応答が送られた時，それがディスクへの読み込み要求に対する応答であればハイパーバイザはデータの復号を行う．応答にはグラント参照が含まれないため，対応する要求を保存しておき，要求からグラント参照を取得する．ディスクへの読み込み要求に対する応答の場合には，グラント参照に対応するシャドーグラントページ上のバッファ領域に格納されたデータを復号しながらゲストグラントページに書き込む．最後に，この応答をゲスト I/O リングにコピーする．使用されなくなったゲストグラントページが解放された場合，同時に対応するシャドーグラントページも解放する．

4.2.4 AES-NI による暗号処理の高速化

UVBond では，CPU の拡張機能である AES-NI をハイパーバイザ内で利用することにより，ディスクの暗号化・復号化処理を高速化する．そのために，WolfSSL の AES-NI 関数を移植した．この関数は XMM レジスタを利用するが，ハイパーバイザ内でこのレジスタを使用すると例外が発生した．Xen では VM の仮想 CPU をスケジューリングする際に XMM レジスタの復元を遅延しており，XMM レジスタへのアクセス時に例外を発生させて復元するためである．そこで，AES-NI 関数を実行する前に CR0 レジスタの TS ビットをクリアしてこの例外が発生しないようにし，実行後に元の状態に戻すようにした．

4.3 完全仮想化ディスク I/O の暗号化

UVBond は準仮想化ディスク I/O だけでなく，完全仮想化ディスク I/O にも対応している．これは，完全仮想化 OS を起動する場合，ゲスト OS が起動する前に実行される BIOS は完全仮想化ディスク I/O を用いるためである．BIOS からのディスク読み込みは，ハイパーバイザにおいて IN 命令をエミュレートすることで実現されており，512 バイト単位での読み込みが行われる．512 バイトの読み込みは 4 バイト分の IN 命令と 508 バイト分の IN 命令の繰り返しに分けてハイパーバイザにトラップされる．AES は 16 バイト単位でしか復号を行うことができないため，後者の 508 バイトの読み込み時に 512 バイト分をまとめてハイパーバイザ内で復号する．VM の起動時には BIOS のディスクへの書き込み機能は使われないため，書き込みの暗号化には未対応である．また，DMA 転送にも未対応である．

4.4 VM の正常起動の確認

UVBond では，ハイパーバイザに登録されたディスク暗号鍵がユーザのものであるかどうかの確認を VM 識別子の送信と同時に行う．起動した VM の VM 識別子を取得するためにドメイン 0 がハイパーコールを発行した際に，ハイパーバイザは VM 識別子に確認用のデータを付加してディスク暗号鍵で暗号化す

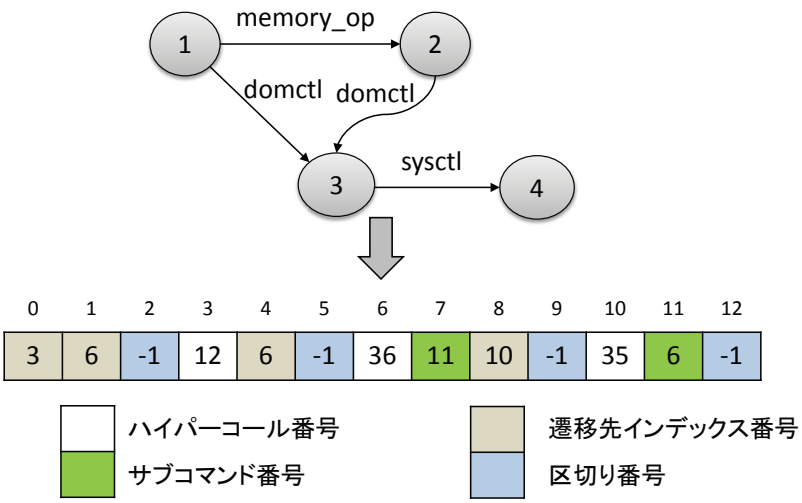

図 9　ハイパーコール列のシリアライズ

る．これを受け取ったクライアントは自身の持つディスク暗号鍵で復号し，確認用データを正しく復号できればユーザの持つ鍵とハイパーバイザに登録されている鍵が同じものであることが確認できる．

さらに，VM の起動に用いられる暗号化ディスクとディスク暗号鍵が正しく対応していることを確認するために，UVBond はディスクのブートセクタのチェックを行う．ブートセクタは VM の起動時に必ず読み込まれ，固定のマジックナンバが格納されている．そのため，ハイパーバイザが復号後にマジックナンバを確認できれば，暗号化ディスクの単純な置き換えを自動的に検出することができる．暗号化ディスクがユーザのものであることを完全に保証するには，マークルハッシュ木などを用いてディスク全体の整合性をチェックする必要がある．しかし，ハッシュデータがディスクイメージとは別に必要となり，ハッシュのデータ量が大きくなるためハイパーバイザ内に登録して用いるのは現実的ではない．このような整合性チェックを行わなくても，正しい暗号化ディスクが用いられているかどうかは VM が正しく起動したかどうかで容易に判断することができる．

4.5　管理コマンドの実行

起動した VM に対して管理コマンドを実行する際には，クライアントは VM 識別子と管理コマンドに対応するハイパーコール列の組を暗号化してドメイン 0 に送信する．ハイパーコール列はオートマトンであるが，暗号化や送信を容易にするために図 9 のように 1 次元の配列にシリアライズして扱う．この配列はある状態に遷移するための入力（ハイパーコール番号）とその状態からの遷移情報の組で構成される．domctl や sysctl などのハイパーコールは様々なサブコマンドを持つため，ハイパーコール番号とサブコマンドを組にして扱う．

ドメイン 0 は受信した VM 識別子とハイパーコール列の組を管理コマンドの実行前にハイパーバイザに渡す．ハイパーバイザはハイパーコール列の登録を行うためにまず，指定された VM に登録されているディスク暗号鍵を用いて VM 識別子とハイパーコール列の組を復号する．そして，復号された VM 識別子と VM に登録されている VM 識別子を比較し，一致していればハイパーコール列をその VM に登録する．VM 識別子が一致しなかった場合，ユーザが意図していない VM へのアクセスを行おうとしていると判断できる．

ハイパーコール列が登録された後，管理コマンドの実行中はハイパーコールのチェックが行われる．チェック対象となるハイパーコールが呼ばれた場合，ハイパーコール列の現在のインデックスが指す状態からの遷移先を調べる．呼ばれたハイパーコールが正しい

入力であれば現在のインデックスを更新して次の状態に遷移し，そのハイパーコールの実行を許可する．ハイパーコール列において正しい入力ではないハイパーコールが呼ばれた場合には，不正な操作と判断し，ハイパーコールの実行を拒否する．管理コマンドの実行終了時に，ハイパーコール列が受理状態になったかどうかの結果をハイパーバイザが暗号化し，クライアントに送信する．

UVBond はハイパーコール列を登録したプロセスにだけそのハイパーコール列を適用する．これは，他の管理コマンド等によって呼び出されたハイパーコールと区別して判定できるようにするためである．ハイパーバイザにおいてプロセスを識別するために，プロセスに固有な CR3 レジスタの値を用いる．CR3 レジスタには OS カーネル内に作成されるプロセスのページディレクトリのアドレスが格納されている．ハイパーコール列を登録する際に CR3 レジスタの値も登録しておき，CR3 レジスタの値が一致する場合だけそのハイパーコール列のチェックを行う．

5 実験

UVBond の有効性を確かめるための実験を行った．実験には，Intel Xeon E3-1290 の CPU，8GB のメモリ，1TB の HDD を搭載したマシンを使用し，ハイパーバイザとして Xen 4.4.0 を用いた．ドメイン 0 では Linux 3.16，ドメイン U では Linux 3.13 を動作させた．ユーザ VM には 2 個の CPU，2GB のメモリ，20GB のディスクを割り当てた．

5.1 管理コマンドの実行

VM 識別子とハイパーコール列を利用して管理コマンドを実行する際に，VM リダイレクト攻撃を含めた不正な操作を検知可能かを確認するための実験を行った．この実験では，VM の一時停止，再開を行う管理コマンドを実行した．実験の結果，操作対象の VM と VM 識別子が対応しており，かつ，ハイパーコール列と管理コマンドが対応している場合だけ正常に管理コマンドが実行されることを確認した．指定した VM に対応しない VM 識別子を指定した場合や，ハイパーコール列に対応しない管理コマンドを実行した場合にはそのことを検知できた．

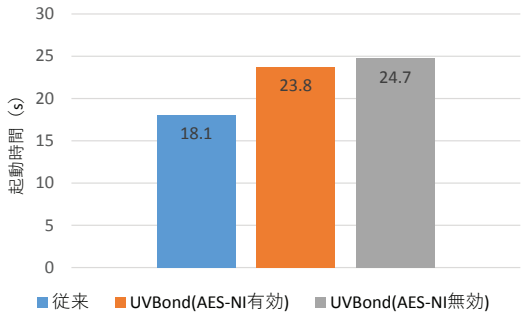

図 10 VM の起動時間

5.2 VM の起動時間

UVBond を用いた場合に VM の起動時間を測定した．測定対象は，UVBond で AES-NI を有効にした場合と無効にした場合，UVBond を用いない従来システムの 3 種類である．ドメイン 0 で起動コマンドを実行してから VM が起動するまでの時間の測定を 5 回ずつ行った．ファイルキャッシュの効果を除外するために，VM の起動毎にドメイン 0 のページキャッシュをクリアして測定を行った．実験結果を図 10 に示す．

起動時間は従来システムが最も短く，AES-NI を有効にした UVBond では 5.7 秒長くなった．これはディスク暗号処理のオーバヘッドに加え，新たに追加した処理によるものだと思われる．AES-NI を無効にした場合と比較すると，AES-NI による高速化は 1.0 秒であることが分かった．

5.3 ディスク I/O 性能

ディスク I/O ベンチマークである fio を用いて性能測定を行った．この実験では 5.2 節で用いた 3 種類のシステムに加え，従来システムにおいて dm-crypt を用いてゲスト OS レベルで暗号化を行う場合についても測定を行った．VM の読み込み性能と書き込み性能を，シーケンシャルアクセスとランダムアクセスそれぞれにおいて 10 回ずつ計測した．結果を図 11，図 12 に示す．

いずれの場合においても，UVBond において AES-

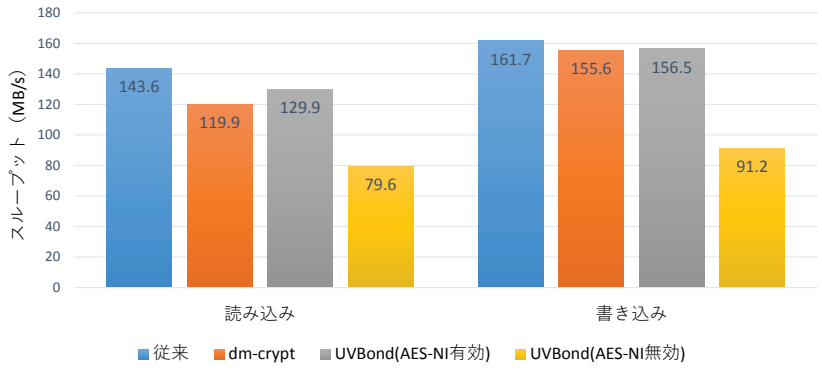

図 11　シーケンシャルアクセス

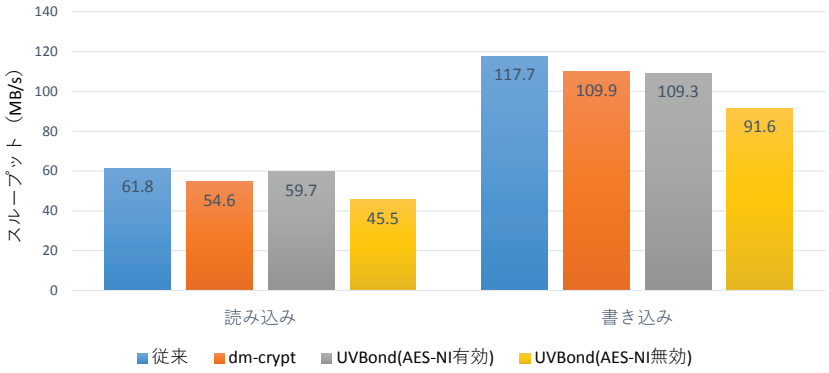

図 12　ランダムアクセス

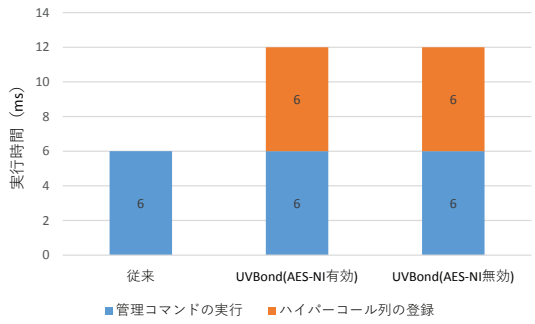

図 13　管理コマンドの実行時間

NI を有効にした場合は無効にした場合と比べて大きく性能が向上した．このことから，AES-NI の高速化の効果があることが分かった．AES-NI を有効にした場合，UVBond は従来システムと比べていずれも 10%以下の性能低下に抑えられている．また，dm-crypt と比べると読み込み性能は少し高くなり，書き込み性能は同等であった．

5.4　管理コマンドの実行時間

ドメイン 0 において管理コマンドの実行時間の計測を行った．測定対象として 5.2 節で用いた 3 種類のシステムを用い，VM の一時停止を行う管理コマンドの実行時間をそれぞれ 10 回ずつ測定した．UVBond においてはハイパーコール列の暗号化およびハイパーバイザへの登録にかかる時間についても測定した．結果を図 13 に示す．

管理コマンドの実行時間はハイパーコール列のチェックを行う UVBond でも従来システムと同等であった．しかし，ハイパーコール列の登録を含めた全実行時間は従来システムの 2 倍となった．これは，用いた管理コマンドの実行時間が非常に短く，ハイパー

コール列の登録の影響が相対的に大きくなったためであると考えられる．ハイパーコール列の暗号化・復号化については，ハイパーコール列が短かったためAES-NI の高速化の効果は見られなかった．

6 関連研究

Self-Service Cloud（SSC）[3] は，クラウドの管理者が干渉できないユーザ専用の管理 VM（Udom0）を提供するため，クラウド管理者はリモート管理の際に VM リダイレクト攻撃を行うことができない．Udom0 のディスク整合性は vTPM を用いて検査され，クライアントとの間は SSL で安全に接続される．ユーザは Udom0 を経由して自身の VM を安全に管理することができる．サービスドメインと呼ばれる VM を用いて安全にディスクの暗号化を行うことも可能である．しかし，vTPM は DomB と呼ばれる VM で動作するため，ハイパーバイザに加えて DomB も信頼する必要がある．また，SSC ではクラウド管理者が VM をまったく管理できなくなる．UVBond ではクラウド管理者が VM を部分的に管理することが可能である．

BitVisor [14] はハイパーバイザ内の準パススルードライバを用いてディスクの暗号化を行うことができる．BitVisor はディスク暗号化に必要な最小限のハードウェアアクセスのみを横取りし，他のアクセスはパススルーすることができるため軽量な VM を実現できる．ATA ホストコントローラ用ドライバでは PIO 転送と DMA 転送に対応しており，シャドー DMA 記述子やシャドーバッファを用いて DMA 転送時の暗号化を実現している．UVBond と異なり完全仮想化 OS にのみ対応しており，準仮想化ディスクドライバを用いることはできない．

CloudVisor [19] はネストした仮想化を用いてハイパーバイザの下で動作するセキュリティモニタにおいて，ディスクの暗号化および整合性チェックを行う．整合性チェックに必要なハッシュデータは管理 VM 経由で提供される．これにより，ユーザが指定したディスクイメージを用いて VM が正常に起動されることが保証される．しかし，ディスク暗号化は外部からの VM のリモート管理とは結び付けられていないため，管理 VM において VM リダイレクト攻撃を行うことが可能である．

FBCrypt [5] は帯域外リモート管理における入出力をハイパーバイザレベルで暗号化することでクラウド管理者への情報漏洩を防ぐ．UVBond と同様に，FBCrypt はフレームバッファを二重化し，暗号化されたフレームバッファをドメイン 0 に提供する．ただし，FBCrypt はページフレーム番号を直接用いたメモリ共有にのみ対応している．また，FBCrypt では I/O リングの二重化は行っていない．キーボード入力については，ハイパーコールを用いて I/O リングへの書き込みを行うことで，入力が復号される前に VM 内のフロントエンドドライバによってアクセスされるのを防いでいる．そのため，ドメイン 0 のバックエンドドライバへの修正が必要である．UVBond では I/O リングを二重化することで，ディスクドライバを修正することなく同期の問題を解決している．

システムコール列を用いた侵入検知システム [6] では，プロセスによって発行されるシステムコールの呼び出し順に基づいて侵入を検知する．プログラムの実際の実行をトレースすることによって正常な実行をシステムコール列として記録しておき，正しいシステムコール列にないシステムコールが実行された場合に侵入と判断している．UVBond のハイパーコール列はこれをハイパーバイザに応用したものである．

7 まとめ

本稿では，VM の暗号化ディスクを介してユーザと VM を強く結びつけることによって VM リダイレクト攻撃を防ぐ UVBond を提案した．UVBond ではハイパーバイザレベルでディスクの暗号化・復号化を行うことにより，ユーザの VM が正しく起動されることを保証する．VM を操作する際は，ハイパーバイザが発行したセキュアな VM 識別子とハイパーコール列を用いることにより，ユーザの指定した管理コマンドがユーザの VM に対して実行されることを保証する．我々は UVBond を Xen に実装し，ディスクを暗号化するオーバヘッドは 10%以下であることを確認した．

今後の課題は，VM が別のホストにマイグレーショ

ンされた時に VM 識別子を使った安全なリモート管理を継続できるようにすることである，また，UVBond のサポートを libvirt に追加し，クラウド管理システムに適用することを計画している．

参 考 文 献

[1] Azab, A., Ning, P., Wang, Z., Jiang, X., Zhang, X., and Skalsky, N.: HyperSentry: Enabling Stealthy In-context Measurement of Hypervisor Integrity, *Proc. Conf. Computer and Communications Security*, 2010, pp. 38–49.

[2] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A.: Xen and the Art of Virtualization., *Proc. Symp. Operating Systems Principles*, (2003), pp. 164–177.

[3] Butt, S., Lagar-Cavilla, H. A., Srivastava, A., and Ganapathy, V.: Self-service Cloud Computing, *Proc. Conf. Computer and Communications Security*, 2012, pp. 253–264.

[4] CyberArk Software: Global IT Security Service, 2009.

[5] Egawa, T., Nishimura, N., and Kourai, K.: Dependable and Secure Remote Management in IaaS Clouds, *Proc. Int. Conf. Cloud Computing Technology and Science*, 2012.

[6] Hofmeyr, S., Forrest, S., and Somayaji, A.: Intrusion Detection Using Sequences of System Calls, *Computer Security*, Vol. 6(1998), pp. 151–180.

[7] 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*, 2015, pp. 443–450.

[8] Li, C., Raghunathan, A., and Jha, N. K.: Secure Virtual Machine Execution under an Untrusted Management OS, *Proc. Int. Conf. Cloud Computing*, 2010, pp. 172–179.

[9] 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(2012), pp. 472–483.

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

[11] PwC: US Cybercrime: Rising Risks, Reduced Readiness, 2014.

[12] Rutkowska, J. and Wojtczuk, R.: Preventing and Detecting Xen Hyper-visor Subversions, *Black Hat USA*, 2008.

[13] Santos, N., Gummadi, K. P., and Rodrigues, R.: Towards Trusted Cloud Computing, *Proc. Workshop Hot Topics in Cloud Computing*, 2009.

[14] T. Shinagawa and H. Eiraku and K. Tanimoto and K. Omote and S. Hasegawa and T. Horie and M. Hirano and K. Kourai and Y. Oyama and E. Kawai and K. Kono and S. Chiba and Y. Shinjo and K. Kato: BitVisor: A Thin Hypervisor for Enforcing I/O Device Security, *Proc. Int. Conf. Virtual Execution Environments*, 2009, pp. 121–130.

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

[16] TechSpot News: Google Fired Employees for Breaching User Privacy, http://www.techspot.com/news/40280-google-fired-employees-for-breaching-user-privacy.html, 2010.

[17] Wang, J., Stavrou, A., and Ghosh, A.: HyperCheck: A Hardware-Assisted Integrity Monitor, *Proc. Int. Symp. Recent Advances in Intrusion Detection*, 2010, pp. 158–177.

[18] wolfSSL Inc.: wolfSSL — Embedded SSL Library for Applications, Devices, IoT, and the Cloud, http://www.wolfssl.com/.

[19] 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*, 2011, pp. 203–216.