Vol.2011-ARC-195 No.15
Vol.2011-OS-117 No.15
2011/4/13

IaaS 環境における VM のメモリ暗号化による情報漏洩の防止

田所秀和[†1] 光来健一[†2,†3] 千葉滋[†1]

Amazon EC2 に代表される IaaS 型クラウドでは利用者に仮想マシン（VM）を提供しているが、自分で管理していないクラウド内のサーバを利用しているためセキュリティが大きな課題となっている。例えば、クラウドの管理者は利用者の VM（ユーザ VM）をサスペンドすることで、容易に VM のメモリから情報を盗み出すことができる。このような情報漏洩を防ぐために、本稿では仮想マシンモニタ（VMM）が必要に応じて VM のメモリ を暗号化するシステム *VMCrypt* を提案する。VMCrypt は、クラウド管理者が管理に用いる VM（特権 VM）からのアクセスに対してだけユーザ VM のメモリを暗号化する。その一方で、サスペンド・レジューム等の VM の管理を行えるようにするために、特権 VM からアクセスする必要があるメモリ領域については暗号化を行わない。

Preventing Information Leakage by Memory Encryption of VMs in IaaS Environments

Hidekazu Tadokoro,[†1] Kenichi Kourai[†2,†3] and Shigeru Chiba[†1]

IaaS such as Amazon EC2 provides users with virtual machines (VMs), but there are security issues because the VMs are hosted in servers of clouds. For example, cloud administrators can easily steal information from users' VMs by only suspending them. To prevent such information leakage, we propose a system called *VMCrypt*, which encrypts user VM memorys. VMCrypt encrypts the memory of user VMs only when the privileged VM accesses it. VMCrypt does not encrypt memory regions that the privileged VM has to access to manage user VMs, such as suspending or resuming them.

1. はじめに

近年、ネットワークを利用してユーザにサービスを提供するクラウドコンピューティングが普及してきている。クラウドコンピューティングはユーザ自身で管理していたハードウェア、ソフトウェア、データ等をネットワーク上からサービスとして提供する。クラウドコンピューティングの一形態である Infrastructure as a Service（IaaS）ではネットワークを通じて仮想マシン（VM）をサービスとして提供する。利用者は必要なときに必要なだけ VM を利用し、サービスの規模を柔軟に調整することができる。

しかしその反面、ユーザは自分で管理していないクラウド内にあるサーバを利用しているためセキュリティが大きな課題となる。クラウド管理者は運用されている VM すべてにアクセスすることが可能であり、各 VM の情報を容易に盗み出すことができる。例えば、クラウド管理者は利用者の VM（ユーザ VM）をサスペンドすることで、VM のメモリ内容をディスクへと保存することができる。その結果、サスペンドによって保存されたメモリイメージを見るだけでクラウド管理者は VM のメモリ上にある情報を盗むことができ、利用者の機密情報が漏洩する恐れがある。ディスクやネットワークと違い、メモリに対してはユーザ VM 内のゲスト OS が暗号化するといった運用で対処することができない。

この問題を解決するために、必要に応じて仮想マシンモニタ（VMM）がユーザ VM のメモリを暗号化することで情報漏洩を防ぐシステム VMCrypt を提案する。VMCrypt はクラウドの管理者が管理に用いる VM（特権 VM）がユーザ VM のメモリにアクセスする時だけユーザ VM のメモリを暗号化することで、メモリの内容を盗み出せないようにする。ユーザ VM が自身のメモリにアクセスする場合は、VMCrypt はメモリの暗号化を行わないため、ユーザ VM は通常通りに正しく動作する。その一方で、サスペンド・レジューム等の VM の管理を通常通りに行えるようにするために、特権 VM からアクセスする必要があるユーザ VM のメモリ領域については暗号化を行わない。

我々は、VMCrypt を Xen 4.0.1 上に実装した。サスペンド・レジュームに関して特権 VM からアクセスする必要があるユーザ VM のメモリ領域は共有情報ページ、P2M テーブル、

†1 東京工業大学
Tokyo Institute of Technology
†2 九州工業大学
Kyushu Institute of Technology
†3 独立行政法人科学技術振興機構, CREST
JST, CREST

Vol.2011-ARC-195 No.15
Vol.2011-OS-117 No.15
2011/4/13

起動情報ページ、ページテーブルの 4 種類であった。これらのメモリ領域の暗号化を行わないことで、正しくサスペンド・レジュームできることを確認した。また、サスペンドによって保存されたメモリイメージからパスワードを盗めないことを確認した。

以下、2 章で IaaS 環境での情報漏洩の問題について述べる。3 章でこの問題を解決する VMCrypt について述べ、4 章でその実装の詳細について述べる。5 章で VMCrypt を用いて行った実験について述べる。6 章で関連研究に触れ、7 章で本稿のまとめと今後の課題について述べる。

2. クラウド管理者への情報漏洩

IaaS 環境では利用者に提供しているユーザ VM の管理者とは別にクラウドの管理者がおり、クラウド管理者がユーザ VM 自体の管理を行っている。Amazon EC2[1] のように Xen を用いて VM を提供している場合、クラウド管理者は特権 VM と呼ばれる VM 管理用の専用 VM を用いてユーザ VM を管理している。例えば、ユーザ VM に提供する仮想ディスクは特権 VM 上のファイルやディスクパーティションを使って実現されており、仮想ネットワークはブリッジで接続されている。また、ユーザ VM のメモリを読み書きする機能を用いて、VM の作成やサスペンド・レジューム、別のホストへのマイグレーションを実現している。

一方で、クラウドの中身がどうなっているかの詳細についてはユーザからは不明であり、クラウド管理者が信頼できるとは限らない。クラウドは複数のデータセンタで構成され、国をまたがって構成されていることも珍しくない。特に、VM を提供している IaaS の場合、VM をどこへでもマイグレーションして実行することが容易なため、VM が実際にどのデータセンタで動いているかは分からない。その結果、セキュリティ意識の低い管理者のいるデータセンタで VM が運用される場合もあるかもしれない。また、システム管理の不備で特権 VM への侵入を許してしまうと、攻撃者にクラウド管理者の権限を奪われる恐れもある。

悪意を持ったクラウド管理者は、特権 VM の機能を用いることで、ユーザ VM から機密情報を簡単に盗むことができる。例えば、仮想ディスクとして使われているファイルを見るだけでディスク内のすべての情報を取得することができ、仮想ネットワークを流れるデータもすべて横取りすることができる。また、ユーザ VM のサスペンドを行い、保存されたメモリイメージを見るだけで容易にメモリ上の情報を盗み出すことができる。このような脅威に対して、ディスクやネットワークの情報はユーザ VM 内で暗号化することで漏洩を防ぐことができる。特権 VM 内のクラウド管理者は仮想ディスクや仮想ネットワークから情報を取得することはできるが、それを復号化することはできない。

しかし、メモリに関してはユーザ VM の中で暗号化するのは難しい。メモリ全体を暗号化してしまうと、ユーザ VM 内のゲスト OS やアプリケーションも動作できなくなってしまうからである。メモリ上にはユーザが入力したパスワードや実行時に生成した暗号鍵などの機密情報が存在しており、クラウド管理者に知られるとセキュリティが保てなくなる。また、メモリ上にはファイルキャッシュが置かれており、ファイルキャッシュを読むことでパスワードファイルや秘密鍵が格納されたファイルなどの情報を盗まれる可能性がある。ユーザ VM 内で暗号化ファイルシステムを使ってディスクを暗号化していたとしても、ファイルキャッシュ上には復号化されたデータが格納される。

サスペンドによって保存されたメモリイメージからの情報漏洩を防ぐために、ユーザ VM のメモリイメージを暗号化して保存する方法が考えられる。これにより、クラウドの管理者が保存されたメモリイメージを盗んだとしても、メモリの内容の漏洩を防ぐことができる。しかし、特権 VM で動作するサスペンドコマンドを信頼しなければならず、悪意ある管理者への情報漏洩を防ぐことはできない。また、VMM がサスペンド機能を提供する方法も考えられる。VMM がユーザ VM のメモリイメージを暗号化してディスクに保存すれば、特権 VM を信頼せずに安全にサスペンドすることができる。しかし、VMM がディスクにアクセスするために、ディスクドライバを持つ必要があり、サスペンド・レジュームのためのコードも必要になるため VMM が肥大化する。

3. VMCrypt

この問題を解決するために、VMM が必要に応じてユーザ VM のメモリを暗号化することで情報漏洩を防ぐシステム VMCrypt を提案する。VMCrypt は、クラウドの管理者が管理に用いる特権 VM からユーザ VM のメモリにアクセスする時だけメモリを暗号化することで、特権 VM からユーザ VM のメモリの内容を盗み出せないようにする。一方、ユーザ VM が自身のメモリにアクセスする場合は、VMCrypt は暗号化していないメモリを見せるため、ユーザ VM は従来通りに正しく動作することができる。VMCrypt は VMM だけで実現され、特権 VM やユーザ VM の OS やアプリケーションには一切変更が必要ない。

VMCrypt はユーザ VM のすべてのメモリを単純に暗号化するわけではなく、一部のメモリ領域は暗号化を行わない。これは特権 VM から従来と同様にユーザ VM の管理を行えるようにするには、一部のメモリ領域の内容を読み書きする必要があるためである。例えば、特権 VM で実行されるサスペンド・レジュームコマンドはユーザ VM のサスペンド・

Vol.2011-ARC-195 No.15
Vol.2011-OS-117 No.15
2011/4/13

レジューム時にユーザ VM のページテーブルを書き換える。ただし、これらのメモリ領域はユーザ VM 内の機密情報ではないため、特権 VM から内容を見ることができても情報は漏洩しない。どのメモリ領域を暗号化するかは VMCrypt が自動的に判別するため、特権 VM であってもユーザ VM の特定のメモリ領域を暗号化しないように設定して意図的に情報を盗むことはできない。

VMCrypt を用いることで、特権 VM からユーザ VM のサスペンドを行う際にはユーザ VM のメモリの内容は自動的に暗号化され、暗号化されたメモリイメージがディスクに保存される。サスペンドコマンドが内容を読み書きする必要があるいくつかのメモリ領域については VMCrypt は暗号化を行わないため、VMCrypt を用いない場合と同様に処理を行うことができる。それ以外のメモリ領域については、サスペンドコマンドが内容解釈する必要がないため、暗号化しても正しくユーザ VM をサスペンドすることができる。また、レジューム時にはレジュームコマンドが暗号化されたユーザ VM のメモリイメージをディスクからユーザ VM のメモリに読み込むと、VMCrypt がそのメモリの内容を自動的に復号化する。レジュームコマンドが内容を読み書きする必要のあるメモリ領域は暗号化されていないため、正常に処理を行うことができる。

3.1 脅威モデル

VMCrypt は特権 VM 内のクラウド管理者がユーザ VM のメモリ上の機密情報を盗む攻撃を想定している。特権 VM からはユーザ VM のディスクやネットワークにもアクセスすることができるが、これらからの情報漏洩はユーザ VM 内で暗号化ファイルシステムや VPN などを用いることで防ぐことができるため、ここでは攻撃の対象にしない。また、ユーザ VM に侵入して情報を盗む攻撃は想定しない。ユーザ VM は十分に管理されており、パスワードやソフトウェアに脆弱性はないものとする。

VMCrypt は VMM で実現されるため、VMM は改竄されていないものとする。VMM の整合性は Remote Attestation によって検証することができる。Remote Attestation とは、別のホストから対象のホストが改竄されていないかを検証する手段であり、TPM などの改竄できないハードウェアの機能を使うことで実現される。TPM を用いてブート時の BIOS や VMM のハッシュ値を第三者が安全に確認することができる。信頼されたブートを検証する Static Attestation だけでなく、Dynamic Attestation[2] を使うことでスタックやグローバル変数などの動的なデータも検証することができる。

ユーザ VM が動作するハードウェアはクラウド管理者によって不正にアクセスされないと仮定する。データセンタではハードウェアをサーバルームに隔離して管理しており、一般の管理者はアクセスすることができないか、アクセスした場合には記録が残ると考えられる。そのため、メモリモジュールからデータを直接見ることで情報を盗む攻撃や、メモリを冷した状態で取り外して別のマシンで見る Cold Boot Attack[3] などは行われないものとする。VMCrypt はユーザ VM のメモリからの情報漏洩のみを対象とし、ユーザ VM のメモリの改竄や破壊は考えない。

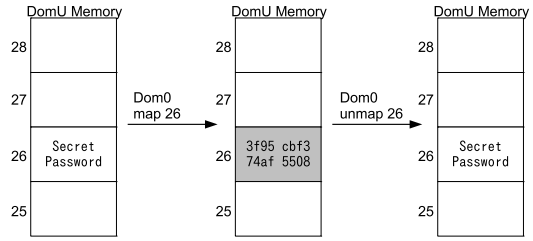


図 1 ドメイン 0 のマップ時に暗号化し、アンマップ時に復号化する様子

4. 実　　装

我々は VMCrypt を Xen 4.0.1 を用いて実装した。Xen においては特権 VM はドメイン 0 と呼ばれ、ドメイン U と呼ばれるユーザ VM を管理している。現在の実装では、ゲスト OS として準仮想化 Linux を対象としている。

4.1 VMM での暗号化・復号化

VMCrypt による VMM によるメモリ暗号化は**図 1** のようにドメイン 0 がドメイン U のページをマップする際に行う。ドメイン 0 がドメイン U のメモリにアクセスするには、必ずドメイン 0 上のプロセスのアドレス空間にドメイン U のメモリをマップする必要がある。このメモリマップの検出は、VMM がページテーブルエントリへの変更を観察することで行うことができる。ドメイン 0 のページテーブルエントリにドメイン U のメモリページに対応するマシンフレーム番号が書き込まれたときに、ドメイン U のページをマップしようとしていると判断し対応するドメイン U のページを暗号化する。ページテーブルへの変更は VMM が管理しており、ドメイン 0 が変更しようとするとハイパーコールやページフォールトによって VMM に制御が移り、必ず監視することができる。

一方、暗号化したページの復号化はマップしたページをアンマップした後に行う。このアンマップの検出も、ドメイン 0 によるページテーブルエントリの書き換えを検出することで行う。書き換え前のページテーブルエントリのマシンフレーム番号がドメイン U のもの

Vol.2011-ARC-195 No.15
Vol.2011-OS-117 No.15
2011/4/13

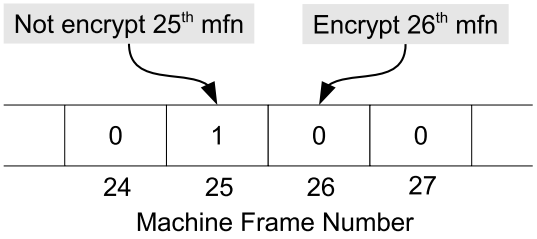


図 2 非暗号ページを表わすビットマップ

だった場合、ドメイン U のページをアンマップしようとしていると判断し、対応するドメイン U のページの復号化を行う。

VMCrypt は、同じページを複数回暗号化しないように、ドメイン U のページがドメイン 0 にマップされている数を管理する。マップされている数が 0 から 1 になったら暗号化し、それ以降のマップでは暗号化しない。また、マップされている数が 1 から 0 になったら復号化する。ドメイン 0 にマップされている数は、ページの情報を管理する page_info 構造体内に管理している。

VMCrypt が用いている暗号化アルゴリズムは AES-256 であり、Linux カーネル 2.6.37 から移植した。暗号化鍵は VMM がメモリ内に保持しており、特権 VM はアクセスすることはできない。現在の実装では、暗号化鍵は VMM のコード中に埋め込んでいる。

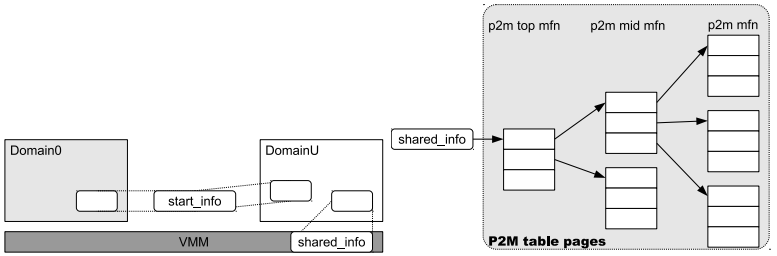
4.2 非暗号化ページ

VMCrypt は、ドメイン 0 がアクセスする必要があるドメイン U のページは暗号化しない。VMCrypt はこのようなページを**図 2** のようなビットマップを用いて管理する。ドメイン 0 がドメイン U のページをマップする際、VMCrypt はこのビットマップを参照し、暗号化すべきかどうかを判断する。もし暗号化しないページならそのままマップを許可し、暗号化するページならば暗号化してからマップする。復号化する時も同様にビットマップを参照して復号すべきかどうかを判断する。

ドメイン U のサスペンド・レジュームを行えるようにするために、共有情報ページ、P2M テーブル、起動情報ページ、ページテーブルの 4 種類を暗号化しないページとする必要がある。

4.2.1 共有情報ページ (shared_info)

図 3(a) のように、shared_info は VMM とゲスト OS が実行中に情報を共有するために



(a) shared_info と start_info　　(b) P2M テーブルの構造

図 3 非暗号化ページ

用いられるページである。VMM とゲスト OS はこのページを使い、仮想 CPU の割り込み通知や時間の情報を共有している。shared_info はドメイン毎に存在するデータ構造であり、ドメインを作る時に VMM が確保し、ドメインを終了するときに解放する。

shared_info には、後述する P2M テーブルの情報が含まれており、サスペンド時にドメイン 0 が読み込み、P2M テーブルにアクセスする。shared_info が存在するページは VMM が管理しているので、VMM は簡単にこのページを暗号化しないようにすることができる。

ビットマップへは、shared_info が作られた時点で登録し、サスペンド時に取り除く。レジューム時には新しいページが割り当てられるので、サスペンド時に取り除く必要がある。

4.2.2 P2M テーブル

図 3(b) のように、P2M テーブルはドメイン U のメモリ上に存在する木構造であり、疑似物理フレーム番号からマシンフレーム番号への対応を管理している。ドメイン U をサスペンドする際、ドメイン U が持つページのマシンフレーム番号をすべて取得するためにドメイン 0 が P2M テーブルを参照する。

VMM は shared_info からこの木構造をたどることで、P2M テーブルに使われるすべてのページを知ることができ、ビットマップに登録することができる。P2M テーブルは 3 段の木構造をしており、この木構造の先頭ノードは一つのページであり、中間レベルノードのページ番号の配列になっている。中間レベルノードは実際に P2M マッピングを管理する葉ノードのページ番号の配列になっている。先頭ノードから順にマップしてアクセスすることで全てのノードのページを知ることができる。

P2M テーブルは、OS の起動時やドメイン U のレジューム直後にドメイン U が作成するため、ドメイン U が起動していない段階ではビットマップに登録することができない。そ

こで、ドメイン 0 が shared_info をマップした時点で VMCrypt が P2M テーブルをたどりビットマップに登録する。P2M テーブルにアクセスするためには最初に shared_info をマップしてアクセスする必要があるため、ドメイン 0 は常に P2M テーブルを非暗号化ページとして扱うことができる。

4.2.3 起動情報ページ (start_info)

図 3(a) のように、start_info はドメイン 0 とゲスト OS が情報を共有するために用いられるページである。このページは、カーネルのブートパラメータやドメインに割り当てられたページの数など、ゲスト OS をブートストラップするための情報が含まれている。start_info には仮想コンソールを実現するためのページのマシンフレーム番号が含まれており、レジューム時にドメイン 0 が新しいマシンフレーム番号を書き込む。

VMM は start_info として使われるページを管理していないが、このページはドメイン U の起動時およびレジューム時にドメイン 0 がドメイン U に通知しており、通知の内容を盗み見ることで VMM は start_info を知ることができる。通知の方法はアーキテクチャ依存であり、x86_64 では仮想 CPU のレジスタを用いている。起動時には rsi レジスタ経由で仮想アドレスが通知され、、レジューム時には edx レジスタ経由でマシンフレーム番号が通知される。VMM はドメイン U の実行を再開する最初の unpause ハイパーコール中でこれらのレジスタをチェックすることで start_info ページを特定し、ビットマップへ登録する。

起動時に指定されるこの仮想アドレスは、疑似物理アドレスへのストレートマッピングになっており、疑似物理フレーム番号を直接求めることができる。疑似物理フレーム番号からマシンフレーム番号を求めるために、一般には P2M テーブルが使われる。しかし、起動時にはまだ P2M テーブルは作られていないため、VMM 内で管理されている M2P テーブルを全探索することで変換を行う。M2P テーブルはマシンフレーム番号から疑似フレーム番号への対応表である。現在の実装では、ページ数に対し二乗の計算量になってしまうため、ページ数が増えるとこの変換にかかる時間が線形よりも速く増加する。

4.2.4 ページテーブル

ドメイン 0 がドメイン U をサスペンドする際、ドメイン U のページテーブルを書き換えてから保存する。準仮想化 Linux では、ページテーブルは仮想アドレスからマシンフレーム番号への対応表であるため、ドメイン 0 はページテーブルのマシンフレーム番号を疑似物理フレーム番号に書き換える。レジューム時にはこの疑似物理フレーム番号を新しく割り当てられたマシンフレーム番号に書き換えることで、正しくレジュームすることができる。

ドメインのページテーブルの変更は、ハイパーコールやページフォールトにより VMM が行うので、VMCrypt は簡単にこのページを認識することができる。VMCrypt はどのページがページテーブルとして使われたかを常に記録しており、あるページがドメイン U のページテーブルとして使われたら、ビットマップへ登録し、使われなくなったらビットマップから登録を削除する。VMM 内ではページ毎にそのページが何に使われるかを表わす型がつけられている。ページテーブルとして使われるページには `PGT_l{1,2,3,4}_page_table` や `PGT_seg_desc_page` という型がつけられており、ページを確保する関数やページを解放する関数でページの型を調べることにより、VMCrypt はページテーブルとして使われるページを知ることができる。

4.3 VMCrypt を用いた起動

VMCrypt では、ドメイン U の起動を特別扱いしてメモリの暗号化は行わず、非暗号化ページをビットマップに登録することだけを行う。現在の実装ではドメイン U の起動かどうかは、ドメイン 0 から手動で知らせている。メモリの暗号化を行わないことで、通常と同じくドメインを起動できる。

ドメイン U を起動するために、最初に VMM がドメイン U を作る。その際に shared_info が作られるので、VMCrypt はそのページを非暗号化ビットマップに登録する。次に、ドメイン 0 がドメイン U のメモリをマップしてカーネルイメージを書き込む。さらに、ドメイン 0 がドメイン U の最初のページテーブルを設定するので、VMCrypt はページテーブルとして使われるページをビットマップに登録する。そして、ドメイン 0 は start_info や shared_info に適切に値を設定し、仮想 CPU 経由で start_info をドメイン U に知らせる。最後に、ドメイン 0 がドメイン U のすべてのページを unmap してから unpause ハイパーコールを呼び、ドメイン U が起動する。unpause ハイパーコール中で仮想 CPU のレジスタを調べることで、VMCrypt は start_info ページをビットマップに登録する。

unpause ハイパーコールを実行した時点からメモリの暗号化が有効になり、ドメイン 0 はドメイン U のメモリにアクセスしても情報を盗むことができない。ドメイン U の実行中にも、VMCrypt はページテーブルとして使われるページの変更によってビットマップを更新し続ける。

4.4 VMCrypt を用いたサスペンド

サスペンドは、ドメイン 0 がドメイン U のメモリをファイルに保存することで行われる。最初に、ドメイン 0 はドメイン U の shared_info をマップしてドメインの情報や P2M テーブルにアクセスする。VMCrypt は shared_info ページを暗号化せずにマップし、同時に P2M テーブルとして使われているページをビットマップに登録しドメイン 0 がアクセス

Vol.2011-ARC-195 No.15
Vol.2011-OS-117 No.15
2011/4/13

できるようにする。

次に、ドメイン 0 がドメイン U のページを順番にマップしながらメモリを読み取り、ファイルに保存していく。このとき VMCrypt はビットマップに従ってドメイン 0 がマップしたページを暗号化する。ドメイン 0 は、ドメイン U のページテーブルエントリを書き換えてから保存する。

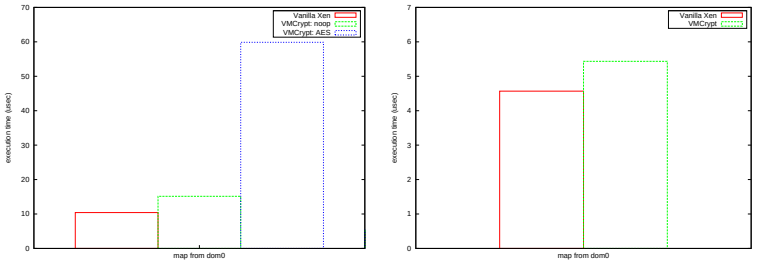
さらに、ビットマップをサスペンド時にドメイン U のメモリに埋め込むことで、レジュームの際に利用できるようにする。埋め込むメモリ領域は、BIOS のメモリマップの設定を表わす e820 を操作し、メモリの一部をゲスト OS に使わせないようすることで確保する。e820 は疑似物理アドレスの範囲とその使い方を表わす対応表になっており、reserved になっているとゲスト OS はそのメモリを使わない。埋め込んだビットマップは VMCrypt によって暗号化されるため、特権 VM がビットマップを改竄することはできない。

4.5 VMCrypt を用いたレジューム

レジュームは、ドメイン 0 がサスペンドイメージをドメイン U のメモリに戻すことで行われる。最初に、VMM がドメイン U を作成し、VMCrypt が shared_info をビットマップに登録する。次に、ドメイン 0 はドメイン U のページを順番にマップしながら暗号化されているページも含めてファイルの内容をメモリにそのまま書き込んでいく。メモリに書き込み後、ユーザ VM のメモリに埋め込んでおいたビットマップを取得する。さらに、ページテーブルエントリを疑似物理フレーム番号から新たに割り当てられたマシンフレーム番号に書き換える。最後にドメインを再開する unpause ハイパーコール中で、ビットマップにしたがってメモリを復号化する。すべてのメモリを書き込むまでビットマップが取得できないため、メモリの復号化はアンマップ時には行うことができない。unpause ハイパーコール内で一括で行うため、疑似物理フレーム番号を順番にマシンフレーム番号に変換し、対応するページを復号化する。この変換には P2M テーブルを使うことができないため、start_info と同様に M2P テーブルを用いて行う。

4.6 Grant Table

Grant Table を使ってドメイン 0 はドメイン U のメモリを読み書きすることができるが、VMCrypt はこのページを暗号化しない。Grant Table はドメイン間でのページ共有やページ移送を行うための機構である。例えば、ドメイン U のフロントエンドドライバとドメイン 0 のバックエンドドライバがデータを受け渡すために用いられている。しかし、Grant Table ではドメイン U が明示的に許可したページしかドメイン 0 は読むことができないため、意図しない情報漏洩が起こる危険性はない。ドメイン 0 に渡されるのはディスクやネットワークのデータであるが、これらはゲスト OS で暗号化することで情報漏洩を防ぐことが可能である。



(a) ドメイン 0 からマップ・アンマップするのにかかる時間　(b) ドメイン U からマップ・アンマップするのにかかる時間

図 4 マップ・アンマップするのにかかる時間

5. 実　　験

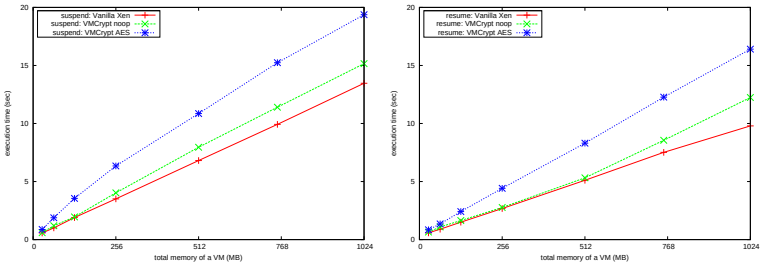
VMCrypt のオーバーヘッドの測定と、VM のメモリからの情報漏洩の防止を確認する実験を行った。実験には、Xeon 2.67GHz 8 コア、メモリ 12GB の PC を使い、Xen 4.0.1、ドメインのカーネルに 2.6.32.27 を用いた。

5.1 非暗号化ページの判定と暗号化のオーバヘッド

ドメイン 0 からドメイン U のメモリをマップ・アンマップする時の非暗号化ページの判定と暗号化のオーバヘッドを測定するため、VMCrypt を用いた場合にドメイン U のページをマップ・アンマップするのにかかる時間を測定した。暗号化方式として AES となにもしない noop の場合について測定した。比較のために、オリジナルの Xen(vanilla Xen) においてメモリをマップ・アンマップするのにかかる時間も測定した。事前にマップ対象の仮想アドレスのページフレーム番号を求めておき、Xen の xc ライブラリの xc_map_foreign_range 関数を用いてマップした。測定は 1 ページのメモリをマップしその後アンマップする操作を 10 万回繰り返して平均を求めた。

図 4(a) が結果である。VMCrypt は vanilla Xen に比べて、暗号化を行わない場合でも 4.7μs 遅くなった。これは 45%のオーバーヘッドである。これは、マップ・アンマップによってページテーブルの書き換えが行われた時、暗号化・復号化すべきかの判定を行うオーバーヘッドである。また、AES を用いると 45μs 遅くなった。これは 295%のオーバーヘッ

Vol.2011-ARC-195 No.15
Vol.2011-OS-117 No.15
2011/4/13



(a) サスペンドにかかる時間　　(b) レジュームにかかる時間

図 5　サスペンド・レジュームにかかる時間

ドである。

5.2　ページテーブルの操作のオーバーヘッド

ページテーブルの操作のオーバーヘッドを測定した。ドメイン U 内でページをマップ・アンマップする操作を 10 万回繰り返し、平均時間を求めた。VMCrypt は、ページテーブルを書き換える時に、マップするページが暗号化対象かをチェックする。同じドメイン内でのマップなので暗号化は行わない。結果は**図 4(b)** である。Vanilla Xen に比べて VMCrypt は 0.87us 遅くなった。

5.3　サスペンド・レジュームにかかる時間

VMCrypt を用いた場合のサスペンドとレジュームにかかる時間をそれぞれ測定した。ドメイン U に割り当てるメモリ量を 26MB から 1024MB まで変化させ、暗号化を行わない Vanilla Xen と比較した。VMCrypt は暗号化方式として AES と何もしない noop を用いた。**図 5(a)** がサスペンド時の結果である。サスペンドがメモリ量に比例した時間がかかることがわかる。1GB の場合、noop では vanilla Xen に比べてサスペンド時間が 13%増加した。これが非暗号化ページの判定にかかるオーバーヘッドである。また、AES の場合 noop に比べてサスペンドの時間が 31%増加した。これがメモリの暗号化にかかるオーバーヘッドである。

図 5(b) がレジューム時の結果である。1GB の場合、noop では vanilla Xen に比べてレジューム時間が 25%増加した。また、AES は noop に比べて 43%レジューム時間が増加した。特に、512MB を越えたところから実行時間が Vanilla Xen の実行時間から乖離しているのは、M2P テーブルを使って疑似物理フレーム番号からマシンフレーム番号を求めるための時間がページ数の二乗に比例して増加するためであると考えられる。

```
root@mach# strings quattro1.img | grep 'root:\$'
acroot:$6$aCJuBx5O$5HqjJyEGM.hDUBnczt2J.j6jN41.GO2kHlNXHZrurOZpqL/Elnbc489
ZrZqLD2gsPDB.yVcK6trNXAquhKFkG0:14879:0:99999:7:::
root@mach#
```

図 6　VMCrypt を使わない場合のシャドウパスワードの内容を取得する攻撃の様子

5.4　パスワード漏洩防止の確認

サスペンドで保存したメモリイメージに対してシャドウパスワードの内容を取得する攻撃を行い、VMCrypt がメモリからの情報漏洩を防げることを確認する実験を行った。まず、起動したドメイン U にコンソールから root でログインすることで、`/etc/shadow` ファイルの内容をキャッシュにのせる。この状態でドメイン U をサスペンドし、キャッシュにのったシャドウパスワードの内容を含んだメモリイメージを保存する。このとき、ドメイン 0 からメモリイメージを読みシャドウパスワードを取得する攻撃を VMCrypt を使う場合と使わない場合で行う。攻撃は strings コマンドで文字列を抽出し、grep コマンドで “root:$” を検索した。“root:$” は root のパスワードのエントリを表わす文字列である。

VMCrypt を使わなかった場合、**図 6** のようにシャドウパスワードの内容が取得できたが、VMCrypt を使った場合には取得できなかった。これにより、VMCrypt を使うことで正しくメモリを暗号化できたと考えられる。

6. 関 連 研 究

Trusted Cloud Computing Platform[4] は、IaaS 環境においてユーザ VM の機密性と一貫性を保証する。Trusted Coordinator が各サイトで信頼できる VMM が動いているかを検査することで、信頼できるサイトにのみ VM を移送できるようにする。さらに、移送時に復元しようとしている VM の状態が改竄されていないかを VMM が検査することで、VM の一貫性を保証する。

Domain Disaggregation[5] は、ドメイン 0 の機能の一部をドメイン B と呼ばれる専用のドメインに移すことでセキュリティを高める。例えば、ドメイン U を作成するコードをドメイン 0 からドメイン B に移すことで、ドメイン 0 はドメイン U のメモリにアクセスする機能を持つ必要がなくなる。しかし、特権 VM の管理者がドメイン B も管理するため、管理者はドメイン B から情報を盗むことが可能である。

Vol.2011-ARC-195 No.15
Vol.2011-OS-117 No.15
2011/4/13

OverShadow[6] は、VMM がプロセスのメモリを暗号化することで、ゲスト OS が乗っ取られてもメモリの機密性と一貫性を保証するシステムである。VMM がゲスト OS 上のコンテキストを認識して、ゲスト OS がプロセスのメモリアクセスする時にページを暗号化する。プロセスに、暗号化されていないメモリを見せることで、プロセスは正しく動作することができる。メモリを暗号化した状態でもシステムコールなど OS の機能を正しく使えるように、専用のプログラムローダを用いる要がある。VMCrypt は Overshadow を特権 VM とユーザ VM に対して適用したと考えることもできるが、ユーザ VM への変更は不要である。

SP3[7] も OverShadow と同様に VMM がプロセスのメモリを暗号化することで、ゲスト OS への情報漏洩を防ぐシステムである。ページ毎に細かいアクセス制御を設定することができ、許可されないプロセスやカーネルがアクセスした場合には、暗号化したページを見せる。暗号化したページと暗号化していないページを両方保持しておき、同時にアクセスできるようにすることで、オーバーヘッドを削減している。一方、VMCrypt はドメイン 0 のメモリアクセス時にはドメイン U を一時停止しているという前提で、メモリを書き換えて暗号化している。

BitVisor[8] は、VMM がゲスト OS とは独立に、ディスクやネットワークを暗号化する。VMM が I/O を監視し、ゲスト OS に対して透過的に I/O パケットを暗号化することでセキュリティを向上させる。BitVisor 暗号化の対象は I/O のみでありメモリの暗号化には対応していない。

VPFS[9] は、VM を用いて信頼できない OS のファイルシステム上に信頼できるストレージを作ることを可能にする。ファイルの内容は信頼できる OS が暗号化することで、信頼できないファイルシステム上に安全に保存する。VPFS はファイルからの情報漏洩を防ぐが、VMCrypt はメモリからの情報漏洩を防ぐ。

7. まとめと今後の課題

本稿では IaaS 環境において、VM のメモリからクラウド管理者への情報漏洩を防ぐシステム VMCrypt を提案した。VMCrypt は、特権 VM がユーザ VM のメモリにアクセスする時に、VMM の中で暗号化を行う。特権 VM がサスペンド・レジュームなどのユーザ VM の管理のためにアクセスする必要があるユーザ VM のページを暗号化しないようにすることで、クラウド管理者は特権 VM から従来通りユーザ VM を管理できる。

今後の課題は、ライブマイグレーションへの対応である。ライブマイグレーション中には、ドメイン U を動かしたままドメイン 0 がメモリアクセスを行うため、単純には対応できない。また、完全仮想化への対応も必要である。完全仮想化では、VRAM や DMA などのためにドメイン 0 がアクセスする多くのメモリ領域を非暗号化ページとして識別する必要がある。

参 考 文 献

1) Amazon Web Services: Amazon Elastic Compute Cloud, `http://aws.amazon.com/ec2/`.
2) Chongkyung Kil, Emre Can Sezer, Ahmed M. Azab, Peng Ning, Xiaolan Zhang: Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence, *Proc. Intl. Conf. Dependable Systems and Network*, DNS '09, pp.115–124 (2009).
3) J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, Edward W. Felten: Lest We Remember: Cold Boot Attacks on Encryption Keys., *USENIX Security Symposium*, pp.45–60 (2008).
4) Santos, N., Gummadi, K.P. and Rodrigues, R.: Towards trusted cloud computing, *Proc. Conf. Hot topics in cloud computing*, HotCloud'09, pp.3–3 (2009).
5) Derek GordonMurray, GrzegorzMilos, S.H.: Improving Xen Security through Disaggregation, *Proc. Intl. Conf. Virtual Execution Environments*, VEE '08, pp.151–160 (2008).
6) Xiaoxin Chen, Tal Garfinkel, E. Christopher Lewis, Pratap Subrahmanyam, Carl A. Waldspurger, Dan Boneh, Jeffrey Dwoskin, Dan R. K. Ports: Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems, *Proc. Intl. Conf. Architectural support for programming languages and operating systems*, ASPLOS XIII, pp.2–13 (2008).
7) Jisoo Yang, Kang G. Shin: Using Hypervisor to Provide Data Secrecy for User Applications on a Per-Page Basis, *Proc. Intl. Conf. Virtual Execution Environments*, VEE '08, pp.71–80 (2008).
8) Takahiro Shinagawa, Hideki Eiraku, Kouichi Tanimoto ,Kazumasa Omote, Shoichi Hasegawa, Takashi Horie, Manabu Hirano, Kenichi Kourai, Yoshihiro Oyama, Eiji Kawai, Kenji Kono, Shigeru Chiba, Yasushi Shinjo, Kazuhiko Kato: BitVisor: A Thin Hypervisor for Enforcing I/O Device Security, *Proc. Intl. Conf. Virtual Execution Environments*, VEE '09, pp.121–130 (2009).
9) Carsten Weinhold, Hermann Härtig: VPFS: Building a Virtual Private File System with a Small Trusted Computing Base, *Proc. European Conf. Computer Systems 2008*, EuroSys '08, pp.81–93 (2008).