Vol.2016-OS-136 No.9
2016/2/29

ベアメタルクラウドにおけるハードウェア保護

東 耕平[1,a)]　竹腰 開[2]　深井 貴明[2]　品川 高廣[1]　加藤 和彦[2]

概要：仮想化技術を用いたクラウドコンピューティングによって提供されるサービスが広く利用されている中、ベアメタルクラウドと呼ばれる新しいサービスが近年注目を集めている。ベアメタルクラウドとはInfrastructure as a Serivce (IaaS) の一種であるが、従来の IaaS がユーザに対して仮想マシンを提供するのに対し、ベアメタルクラウドは物理マシンを提供する。これによって、ユーザは安定して高い性能・機能を発揮するマシンを比較的安価かつ容易に利用することが出来る。しかし物理マシンをユーザにそのまま提供してしまうと、ユーザは物理マシンのハードウェアに直接アクセスすることが可能になるため、重要な情報が記憶されている EEPROM などの不揮発性領域のデータが書き換えられてしまうことによって、マシンが恒久的に起動しなくなったりマルウェアに感染して次のユーザが影響を受けるなどの問題が発生する可能性がある。本稿では、軽量なハイパーバイザを利用して、重要なデータが記憶されている不揮発性領域へのアクセスを遮断する手法を提案する。これにより、ベアメタルクラウドの利点は維持しつつも、クラウド事業者が物理マシンのハードウェアを保護して安全性を確保できるようにする。

1. はじめに

近年、仮想化技術を用いたクラウドコンピューティングによって提供されるサービスが広く利用されている。仮想化技術とはハードウェアのリソース（CPU やメモリ、ストレージなど）を、物理的な構成にとらわれずに論理的に分割・統合することができる技術である。仮想化技術を用いると、ユーザの要求に応じてオンデマンドで仮想マシンを作成して提供したり、負荷が高いサーバ上で動作する仮想マシンを別のサーバに移動したりすることが容易になる。クラウドコンピューティングの中でも Infrastructure as a Service (IaaS) と呼ばれるサービスは、データセンター内のサーバ上で動作する仮想マシンをインターネットを通じてオンデマンドでユーザに貸し出すサービスである。

一方、クラウドコンピューティングにおいてベアメタルクラウドと呼ばれる新しいサービスが近年注目を集めている。これは IaaS の一種ではあるが、仮想マシンの代わりに物理マシンのインスタンスをインターネットを通じてユーザに貸し出すサービスである。仮想マシンではなく物理マシンを提供する利点としては、以下の様なものがある。まず第一に、性能のばらつきが無い点である。従来の IaaS では、同一サーバ上の他のユーザによるリソースの利用状況によって仮想マシンの性能に変化が生じてしまうのに対し、ベアメタルクラウドではユーザが物理マシンを専有して利用するため、性能のばらつきが発生しない。第二に、仮想化による性能低下が無い点である。従来の IaaS ではハードウェアの上に仮想マシンモニタやハイパーバイザと呼ばれる仮想化を実現するソフトウェアがあり、仮想化の処理によって一定のオーバーヘッドが避けられない [1]。一方、ベアメタルクラウドでは仮想化の処理がないため物理マシンの性能を最大限活かすことができる。他にも、同一サーバ上で動作する仮想マシン間での情報漏洩がないといったセキュリティ上の利点や、ハードウェアの提供する機能を最大限活用できるといった機能上の利点もある。このように、ベアメタルクラウドは安定して高い性能・機能を発揮するマシンを比較的安価かつ容易に利用したいユーザにとって、従来の IaaS よりも便利なサービスである。

しかし、物理マシンをユーザにそのまま貸し出してしまうと、ユーザがその物理マシンの全てのハードウェアに直接アクセスすることが可能になってしまうため、場合によっては物理マシンの状態を完全にもとに戻すことが簡単にはできなくなってしまう。これは、物理マシンの多くのハードウェアはリセットや電源断などで状態を元に戻すことができるものの、なかには設定情報を保存するための Electrically Erasable Programmable Read-Only Memory (EEPROM) と呼ばれるデバイスや、更新可能なファームウェアを格納するためのフラッシュメモリなど、不揮発性を持ったデバイスが存在しているためである。このような

[1] 東京大学
The University of Tokyo
[2] 筑波大学
University of Tsukuba
[a)] azuma@os.ecc.u-tokyo.ac.jp

不揮発性を持ったデバイスの内容をユーザに書き換えられてしまうと、その状態が物理マシンを返却後も引き継がれてしまうため、クラウド事業者や次のユーザにとって深刻なセキュリティ上の問題が発生する。例えば、設定情報を保存した EEPROM の内容が破壊されてしまうと、マシンによっては BIOS の初期化時にエラーが発生するなどしてマシンが起動しなくなってしまう可能性がある。また、Network Interface Card (NIC) の EEPROM に保存されている MAC アドレスを書き換えられると、ネットワーク上の別のマシンになりすまされて、データの送受信を妨害されたり情報が漏洩・改竄されたりする可能性がある。ファームウェアが格納されたフラッシュメモリ領域が書き換えられた場合には、マルウェアに感染して次にそのマシンを使うユーザの情報が危険にさらされる可能性もある。

BIOS のコードや UEFI のファームウェアが格納されるフラッシュメモリ領域に対しては、近年はハードウェアによる保護や署名などによって不正な書き換えを防止する仕組みが導入されている [2]。しかし、様々な手法によりハードウェアによる保護を回避してソフトウェア的にフラッシュメモリを書き換える攻撃手法も多数知られている [3]。ソフトウェアによりファームウェアの一貫性を検査する手法なども提案されているが [4]、そもそも書き換えられることを防止することは容易ではない。また、設定を保存する EEPROM などの領域に対しては、ハードウェアによる保護はあまりかけられておらず、OS カーネルなどスーパーバイザ権限（Intel CPU における ring 0 など）を持ったコードであれば、容易に書き換えられることが多い。

本研究では、軽量なハイパーバイザを用いて、重要なデータが記憶されている不揮発性領域へのアクセスを遮断する手法を提案する。提案方式では、BIOS のファームウェアや NIC の EEPROM などの不揮発領域への書き込みをおこなう I/O アクセスを特定し、ハイパーバイザを用いてそれらのアクセスを捕捉して無効化する。これにより、OS カーネルのようなスーパーバイザ権限をもったコードであっても、不揮発性領域を書き換えることが出来ないようにする。書き込みが必要になった時は、クラウドの事業者のみがハイパーバイザによる保護を解除出来るようにする。一方で、不揮発性領域へ書き込む I/O アクセス以外のハードウェアへのアクセスは全てパススルーとすることによって、ベアメタルクラウドの利点である仮想化のオーバーヘッドがないという点を可能なかぎり維持する。

本研究の実装では、軽量ハイパーバイザとして BitVisor [5] を用いた。BitVisor とは準パススルー方式を導入したハイパーバイザである。準パススルー方式とは、デバイスを仮想化せず、OS からハードウェアへのアクセスをできるだけパススルーし、セキュリティに関する最小限のアクセスのみを捕捉するアーキテクチャである。BitVisor をベースとして、基本的には全てのハードウェアアクセスがパススルーになるように設定しつつ、不揮発性領域への書き込みをおこなう I/O 命令を捕捉する実装をおこない、書き込みがおこなわれないようにする。

今回の実装では、コンセプト実証として、Intel 社の NIC である I218-V を対象とし、EEPROM へのアクセスを遮断した。実験の結果、ethtool を使って EEPROM に格納された MAC アドレスを書き換えることを防止できることを実際に確認した。また、性能に対するオーバーヘッドは非常に少ないことを確認した。

以降、2 章で研究の動機、3 章で設計、4 章で実装、5 章で実験、6 章で関連研究、7 章で結論を述べる。

2. 脅威モデルと攻撃の例

2.1 ベアメタルクラウドの特性と脅威モデル

ベアメタルクラウドは IaaS の一形態であり、仮想マシンの代わりに物理マシンをユーザに提供する。物理マシンを提供することで、ユーザは他のユーザや仮想化ソフトウェアの影響を受けない他、GPU, Infiniband, NVM-Express ストレージなど高性能なハードウェアを最大限活用できる。ベアメタルクラウドの用途としては、高負荷な Web サービスのバックエンドや科学計算のための計算ノードなどがある。近年では、IBM [6], Internap [7], Rackspace [8] など多くの事業者が実際にベアメタルクラウドのサービスを提供している。

ユーザに対して仮想マシンを提供する種類の IaaS では、一般にユーザは物理的なハードウェアに直接アクセスできず、仮想マシンモニタが提供する仮想的なハードウェアにのみアクセスできる。仮想的なハードウェアは、仮想マシンが作成される度に仮想マシンモニタによってソフトウェア的に作成され、仮想マシンが破棄されると同時に仮想的なハードウェアも破棄される。この為、仮想マシンではユーザが変わる毎にハードウェアやマシンのが使いまわされる事は基本的には無いと言え、仮想マシンのユーザが他のユーザの利用状況等を考える必要は無い。対して、ベアメタルクラウドでは、同じ物理的なハードウェアを別々のユーザが使いまわす事になる。ベアメタルクラウドでは、ユーザが変わる度に事業者が HDD 等の初期化を行う為、見かけ上はその都度新しい環境が作られているように見える。しかしながら、使用している物理的なハードウェアは前のユーザが使っていたものと同じであり、完全に新しい環境が作られているわけではない。全てのユーザが物理的なハードウェアにソフトウェアから直接アクセスできる為、ソフトウェアからハードウェア内部の設定やファームウェアを変更できれば、他の利用者の環境にまで影響を及ぼせる。

以上の特性から、ベアメタルクラウド特有の脅威モデルが 2 つ考えられる。一つ目は事業者に対するサービス拒否攻撃 (DoS) である。物理的なハードウェアの内部で記憶し

ている設定情報等を書き換える事で、ハードウェアを故障させたり誤動作させたりする事が可能である。これは悪質なサービス拒否攻撃（DoS 攻撃）と言え、ハードウェアの交換費用とサービスの停止による損失等が発生する。二つ目は、ハードウェア内部のファームウェア等を書き換えて細工したプログラムを書き込む事でハードウェアに不正な動作をさせる攻撃である。このような攻撃は、攻撃者による不正アクセスのために行われる可能性があり、より多大で広範な損害が発生し得ると考えられる。本研究では、このような攻撃を防止するための手法を提案する。

2.2 物理的なハードウェアを利用した攻撃の例

物理マシンのハードウェアの多くは、EEPROM やフラッシュメモリなどの不揮発性メモリを持っている。例えば、NIC が備える設定情報を記憶する為の EEPROM や、マザーボードが備える BIOS プログラムを記憶する為のフラッシュメモリがある。通常、これらの不揮発性メモリには製造者が意図した仕様通りの値が書き込まれており、それによってハードウェアとソフトウェアが正常に動作するように作られている。NIC であれば、不揮発性メモリにはその NIC の仕様に沿って動作設定や MAC アドレスが書き込まれており、NIC のハードウェアやデバイスドライバはこの値を参照しながら動作する。マザーボード上の不揮発性メモリであれば、コンピュータの起動シーケンスの中で BIOS のプログラムコードを読み込む際に参照される。

このような不揮発性メモリに対しては、仕様上では意図されていないデータを書き込んでハードウェアが誤動作するようにしたり、細工されたプログラムを書き込んでおいてハードウェアやソフトウェアに不正な挙動をさせるといった攻撃がおこなわれる可能性がある。例えば、Intel 社の製造する PRO1000 シリーズの NIC では、動作設定等を記憶する為に EEPROM の不揮発性メモリを搭載している。この EEPROM にはハードウェアもソフトウェアも参照する重要な設定情報が格納されており、EEPROM のデータを誤って破壊した結果 NIC が動作しなくなった例が複数報告されている [9], [10], [11]。

著者らによる実験では、同一シリーズの NIC 二機種に搭載されている EEPROM の全面に対してそれぞれランダムなデータを書き込んだ所、それぞれデバイスドライバによるハードウェア初期化の最初期段階でハードウェアが反応を返さないようになり、ハードウェアとして故障した事が確認された。その内、Lenovo 社のラップトップである ThinkPad X1 上にオンボード搭載された WG82579LM の EEPROM データを破壊したケースでは、破壊した後にコンピュータそのものが全く起動しなくなる事象を確認した。コンピュータが完全に起動しなくなったので原因は究明できていないが、考えられる原因としては、コンピュータそのものの初期化処理の途中で NIC に関係した処理を行うため、NIC が壊れた事で初期化処理が途中から進まなくなった等の可能性が考えられる。

また、細工したプログラムを不揮発性メモリに書き込む攻撃の例としては、研究者によって開発された Broadcom 社製 NIC 内で動作するルートキット [12]、イタリアのセキュリティ企業 Hacking Team が開発した UEFI ルートキット [13] や、アメリカ国家安全保障局（NSA）が開発したとされる「DEITYBOUNCE」 [14], [15] が挙げられる。研究者によって開発された Broadcom 社製 NIC の内部で動作するルートキットは、NIC 上の EEPROM に書き込まれているファームウェアイメージに不正なプログラムを埋め込んで NIC 上で動作させ、攻撃者がネットワーク越しに物理メモリを読み書きする機能を提供する。Hacking Team の UEFI ルートキットは、Intel 製マザーボードに対する偽のアップデータとして動作し、UEFI の不揮発性メモリに独自の不正なプログラムを埋め込む。このプログラムは、UEFI から Windows のシステムディスクを書き換えてバックドアプログラムをインストールする。DEITYBOUNCE は、Dell 製コンピューターの BIOS に対して不正なプログラムを埋め込み、定期的な任意コードの実行を可能にする。

3. 設計

本章では、軽量ハイパーバイザを用いて、仮想化のオーバーヘッドを最小限に抑えつつ、不揮発性メモリへの書き込みを防止する手法の設計について述べる。

3.1 提案手法の概要

ベアメタルクラウドにおけるハードウェア保護の設計においては、以下の 2 つの条件を考慮する必要がある。

- OS から独立したレイヤでハードウェア保護を実現する
- 物理マシンの性能を可能な限り落とさない

ベアメタルクラウドにおいては、OS はユーザが自由にインストールして実行することが出来るため、クラウド事業者にとっては基本的には信用することが出来ない。従って、不揮発性メモリへの書き込みを防止する機構を実現するためには、OS よりも下のレイヤで実現することが不可欠となる。一方で、ベアメタルクラウドにおいては物理マシンの性能を最大限引き出せることが重要であるため、性能が著しく低下するような保護機構を用いることは避けなければならない。

提案手法では、これらの条件をみたすために、軽量なハイパーバイザを用いる。ハイパーバイザは、ゲスト OS よりも高い特権レベルで独立して機能するため、たとえ信用できないゲスト OS が動作していても、ハードウェアの保護機構を確実に実現することが出来る。このことを利用して、不揮発性メモリへの書き込みアクセスをハイパーバイザで捕捉することにより、不正な不揮発性メモリの書き換えを確実に防止する。一方で、ベアメタルクラウドの利

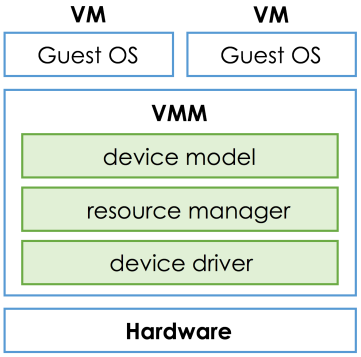

図 1 従来の一般的なハイパーバイザの構造

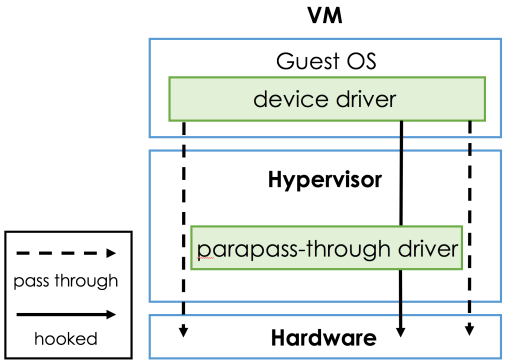

図 2 準パススルー型ハイパーバイザ

点を維持するために、従来のハイパーバイザとは異なり、同時に動作する OS は一つだけに限定する。また、ハードウェアの仮想化は極力おこなわず、ほとんどのハードウェア・デバイスへのアクセスをパススルーとする。これにより、仮想化のオーバーヘッドを可能なかぎり削減して、ベアメタルの性能に近づけることが出来る。

3.2 ハイパーバイザの構造

ハイパーバイザを用いたセキュリティの向上については様々な提案がなされている [16], [17], [18], [19], [20], [21]。これは、ハイパーバイザを用いることで OS に依存せずに強力なセキュリティを実現できるためである。一方で、ハイパーバイザ自身にもセキュリティ上の脆弱性が数多く見つかっており、ハイパーバイザ自身のセキュリティ向上も重要な課題である。ハイパーバイザのセキュリティを向上させるためには、そのコード量を削減することが有効な手法であることが知られている [22]。

図 1 に Xen や VMware などサーバで使用される一般的なハイパーバイザのアーキテクチャを示す。このアーキテクチャでは、複数の仮想マシンを生成するためのデバイスモデルや仮想マシン間の資源管理をおこなうリソースマネージャ、ハードウェアを制御するデバイスドライバなどが必要で、そのコード量が非常に多くなっている。

一方、準パススルー型と呼ばれるアーキテクチャでは、同時に動作する OS を一つに制限して、ハードウェアへのアクセスを原則としてパススルーにしつつ、必要最小限のハードウェアアクセスのみを捕捉することにより、そのコード量を大幅に削減することが可能になっている（図 2）。また、準パススルー型アーキテクチャでは、ハードウェア・デバイスを仮想化しないため、仮想化によるオーバーヘッドを大幅に削減することが出来る。例えば、割り込みコントローラを仮想化しないため、割り込みをソフトウェアでエミュレーションする必要が無い。また、ハイパーバイザに制御が渡る機会が最小限しか無いため、コンテキストスイッチのオーバーヘッドも低く押さえることが出来る。

提案手法では、この準パススルー型のアーキテクチャを踏襲することで、そのセキュリティ上及び性能上の利点を活かした設計としている。

3.3 不揮発性メモリの書き込み保護

EEPROM などの不揮発性メモリは、データの書き込みをおこなうためには、通常より高い電圧をかけるなどによって、いったんその内容を消去する必要がある。そのため、通常の DRAM などとは異なり、物理アドレス空間にマッピングされている領域に対して直接値を書き込むことは出来ない。従って、不揮発性メモリに値を書き込むためには、特殊な I/O 命令を用いて間接的にアクセスする構造になっている場合が多い。

不揮発性メモリへのアクセスの方法は、デバイスごとに固有の方式となっているが、例えば以下の様な手順でおこなわれる。

(1) 書き込み先のアドレスをレジスタなどで指定する
(2) 書き込む値をレジスタなどに格納する
(3) 書き込み要求をおこなう

それぞれの手順は、PIO (Programable I/O) や MMIO (Memory-Mapped I/O) などの I/O 命令を用いておこなわれるため、ハイパーバイザを用いることでそれぞれの I/O 命令を捕捉することが可能である。 提案方式では、(3) の I/O 命令を捕捉して、書き込み要求がエラーになったように見せかけることで、不揮発性メモリへの書き込みを防止する。また、必要に応じて (1) 及び (2) の I/O 命令も捕捉することで、特定の領域だけ書き込みを禁止することも可能である。

具体的に捕捉をおこなう I/O 命令の内容や、エラーに見せかける方法は、各デバイスの仕様書などを参照して、デバイスごとに実装をおこなう。

3.4 ハイパーバイザの起動

提案手法では、ユーザがインストールするゲスト OS よりも先にハイパーバイザが起動する必要がある。そこで BIOS や UEFI の設定により起動順序を適切に設定して、ハイパーバイザが格納された起動デバイスから優先して起動するようにする。また、ハイパーバイザへの不正アクセ

表 1 EEPROM アクセスに主に関係する SRI レジスタ

HSFC	Hardware Sequencing Flash Control
HSFS	Hardware Sequencing Flash Status
FADDR	Flash Adress
FDATA	Flash Data

スを防止するために、ゲスト OS からハイパーバイザが格納されたデバイスへのアクセスを禁止する。これには、ハイパーバイザが格納されたデバイスを丸ごとゲスト OS から隠蔽したり、適切なアクセス制御をおこなうことによって実現する。

4. 実装

本章では、不揮発性メモリの書き換えを防止する例として、ASRock Z97 Extreme6 マザーボードにオンボードで搭載されている Intel Pro/1000 NIC の一種である I218-V の EEPROM への書き込みアクセスを遮断する実装について述べる。実装には、準パススルー型のハイパーバイザである BitVisor をベースとして用いた。

I218-V の EEPROM へアクセスするためのインターフェースは二種類存在する。一つは NIC である I218-V 上のレジスタであり、もう一つはチップセットである I/O Controller Hub (ICH) 上のレジスタである。どちらを用いてアクセスするかは OS によって異なるが、今回の実験で用いた Linux では、チップセットの Intel ICH8 上の Serial Peripheral Interface (SPI) [23] と呼ばれるレジスタを用いて EEPROM にアクセスしていたので、本章ではこのインターフェイスに対する実装を示す。

SPI レジスタにおいて主にアクセスに使用されるレジスタを表 1 に示す。

HSFC アクセス方法 (Read, Write, Erase)、アクセス開始、アクセスサイズを指定するレジスタ

HSFS アクセス完了やアクセスでエラーが発生したか否かを示すレジスタ

FADDR EEPROM のアドレスを指定するレジスタ

FDATA 書き込みするデータや読み出したデータを格納するレジスタ

EEPROM へのアクセスは以下のような手順で行なわれている

(1) FADDR にアクセスするアドレスを指定
(2) FDATA に書き込む (読み取る) データを格納
(3) HSFC でアクセス方法（Write, Read, Erase）を指定
(4) アクセス要求を発行
(5) HSFS でステートを確認
(6) ステートが DONE なら終了

本実装では、BitVisor を用いて (3) の段階で ICH8 上のレジスタへのアクセスである場合は I/O 要求を捕捉する。ここで I/O の内容を確認し、アクセスが Write や Erase であった場合には I/O 要求を遮断し、(4) の処理が行われないようにする。このとき、I/O 要求を遮断しただけだと、(5) の段階で HSFS のステートが DONE にならないため、処理が終了せずに OS が停止してしまう場合がある。これを防ぐため、(3) で I/O 要求を遮断した際に、同時に HSFS のステートを DONE かつ ERROR に設定する。

表 2 BitVisor を導入したマシン環境

CPU	Intel Core i7-4790K(4.000GHz)
Memory	16GB
Kernel	Linux 3.16.0-38
Distribution	LinuxMint 17.2
NIC	Intel Pro/1000 I218-V

表 3 KVM の詳細

Kernel	Linux 3.16.0-4
Distribution	Debian 8.2
QEMU	2.50

表 4 ネットワーク通信に使用したマシンの環境

CPU	Intel Core i7-3770K(3.500GHz)
Memory	16GB
Kernel	Linux 3.16.0-4
Distribution	Debian 8.2
NIC	Intel Corporation 82579V

5. 実験

本章では、BitVisor による EEPROM の書き換え要求の遮断と、BitVisor を起動させることで生じるオーバーヘッドについて述べる。本実装ではネットワークに関するレジスタへのアクセスに対して新たな処理を加えているため、ネットワーク通信におけるオーバーヘッドを計測した。

5.1 実験環境

BitVisor を使用して EEPROM への書き込みを遮断する実験をおこなったマシンの環境を表 2 に示す。また、比較のため、同一マシンに KVM もインストールして実験をおこなった。KVM の環境を表 3 に示す。

ネットワーク通信に関するベンチマークテストでは、マシン間でのネットワーク通信速度を計測するために、もう一台のマシンを用意した。このマシンには仮想マシンモニタなどはインストールされていない。このマシンの環境を表 4 に示す。

5.2 EEPROM の書き換えアクセスの遮断

本実験では、EEPROM の書き換えの例として、ethtool による MAC アドレスの書き換えアクセスをおこなった。MAC アドレスはネットワーク上でのマシンの特定のために用いられており、NIC 上の EEPROM に記憶されている。ethtool は NIC の設定に用いられるソフトウェアであり、

図 3 MAC アドレスの読み取り

```
$ethtool -e eth1
Offset      Values
------      ------
0x0000:     d0 50 99 2f 51 f2 01 08 ff ff
```

図 4 MAC アドレスの書き換え

```
$ethtool -E eth1 magic 0x15a18086 offset
-0x05 value 0x00
$ethtool -e eth1
Offset      Values
------      ------
0x0000:     d0 50 99 2f 51 00 01 08 ff ff
```

図 5 BitVisor による MAC アドレス書き換えアクセスの遮断

```
$ethtool -E eth1 magic 0x15a18086 offset
-0x05 value 0x00
Cannot set EEPROM data:
Operation not permitted
```

イーサネットデバイスのクエリや制御に用いられる [24]。本実験では、ethtool を用いて MAC アドレスの書き換えを試み、この書き換えアクセスを BitVisor で捕捉・遮断した。

ethtool では、図 3 のようにコマンドを実行することで、MAC アドレスを読み取ることができる。MAC アドレスは 48 ビットであり、6 オクテットで区切りで表されている。また、図 4 のようにコマンドを実行することで、MAC アドレスを変更できる。この場合は 6 オクテット目を f2 から 00 へ変更している。変更された MAC アドレスは EEPROM に記憶されいるため、OS をリブートしても元のアドレスには戻らない。

この書き換え操作に対して、提案手法の実装を用いることで、図 5 のようにアクセスが遮断されることを確認した。表 2 の実験環境では、NIC の EEPROM へのアクセスのインターフェースは、4 章で述べたようにチップセットの ICH8 のレジスタを使用する。提案手法では、ICH8 を経由した I218-V の EEPROM への書き込みを遮断しているため、 EEPROM 上に記憶されている MAC アドレスの書き換えが防止された。この時 HSFS レジスタのステートは DONE かつ ERROR となるようにしているため、ethtool からの ioctl 要求に対して Linux カーネル内のデバイスドライバが EPERM を返し、それを受けて ethtool コマンドがターミナルに”Cannot set EEPROM data: Operation not permitted”と表示している。

5.3 ネットワーク通信におけるオーバーヘッドの計測

BitVisor を用いて EEPROM への書き込みを遮断したマシンにおけるネットワーク通信のオーバーヘッドを計測した。表 2 のマシンをマシン A とし、表 4 のマシンをマシン B とする。マシン AB 同士を LAN ケーブルで直結して、その間のネットワーク通信のスループットとレイテンシを計測した。測定にはベンチマークソフトウェア Netperf (Ver2.7) [25] を使用した。レイテンシの測定には Netperf 内の omnitest を使用した。計測の条件として、測定時間は 1 秒、プロトコルは TCP で行った。

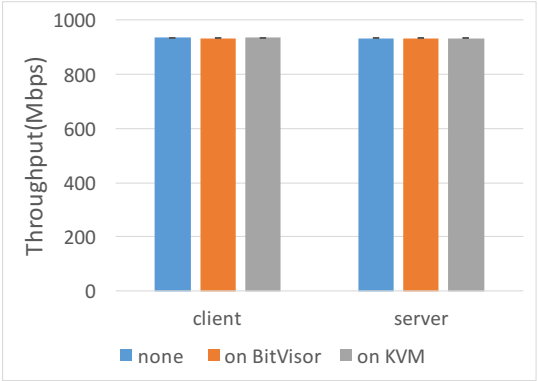

図 6 マシン AB 間のスループット

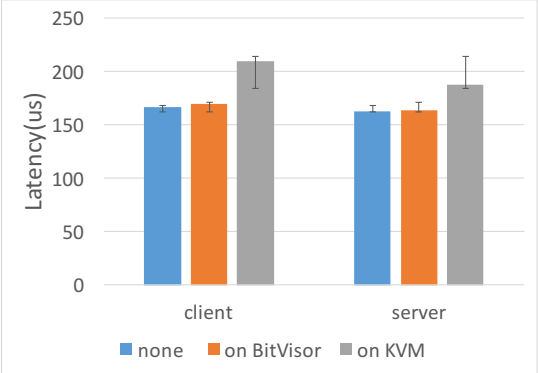

図 7 マシン AB 間のレイテンシ

実験はマシン A で Linux を (1) 直接動かす場合 (”none”)、(2) 提案方式を実装した BitVisor 上で動作させた場合 (”on BitVisor”)、(3)KVM 上で動作させた場合 (”on KVM”) の 3 種類に対して、それぞれマシン A が Netperf の (a) クライアントになる場合 (”client”)、(b) サーバになる場合 (”server”) の 2 種類の組み合わせで、計 6 種類の実験を行った。それぞれ 10 回測定をおこなってスループットとレイテンシの平均値を計算したものを図 6 及び 7 に示す。スループットの単位は Mbps、レイテンシの単位は μs である。

スループットに関しては、図 6 に示すようにハイパーバイザによる影響はほとんど見られなかった。一方、レイテンシに関しては、KVM 上では送信で 40μs、受信で 25μs 程度レイテンシが増加するのに対し、BitVisor 上では送信は 3μs、受信は 0.2μs 程しか増加していない。実験結果から、提案方式では仮想化のオーバーヘッドを非常に低く抑

えてベアメタルクラウドの利点を保ちつつ、EEPROM の書き換えを防止してハードウェアを確実に保護できることが確認された。

6. 関連研究

Loïc [26] らは、NIC の脆弱性を利用することでコンピュータの制御を奪えることを示し、対策として NIC 内のマルウェアを検出する NAVIS を提案している。NAVIS では NIC のファームウェアの一貫性を確認することでマルウェアの検出をおこなうことができるが、この手法では感染を事前に防ぐことは難しい。提案手法では、コンピュータから不揮発性メモリを書き換える手法に対しては、マルウェアの感染を事前に防ぐことが可能である。

Yanlin [27] らは、ホスト OS と周辺機器間で時間制限を設けたチャレンジレスポンスプロトコルを用いることにより、周辺機器の安全性を確認する手法である VIPER を提案している。この手法ではマルウェアに感染した周辺機器にチャレンジレスポンスを送ると計測可能な遅延が発生し、この遅延から周辺機器内のマルウェアの有無を判定することができる。これにより従来では特定できなかったプロキシアタックも含め、周辺機器に対する当時存在したソフトウェア攻撃を全て検出することに成功した。しかし VIPER や NAVIS によるマルウェア検出においては OS が信頼できることが前提となっている。本手法では OS と独立して機能するハイパーバイザを用いることで、信頼できない OS に対してもハードウェア保護を実現することができる。

Fengwei [28] らは、高速に周辺機器の I/O 構成とファームウェアの安全性を確認する IOCheck を提案している。これは x86 アーキテクチャの CPU における System Management Mode (SMM) を利用した手法であり、VIPER や NAVIS と異なり OS が信頼出来なくても良い。この手法では SMM をセットアップする BIOS が安全にブートがされることが必要だが、Static Root of Trust for Measurement (SRTM) により容易に実現できるため、信頼性の高い手法と言える。提案手法を用いると BIOS への書き込みも完全に遮断することができるため、更に安全に IOCheck を行うことができると推測される。

7. 結論

本研究では、ベアメタルクラウドにおけるハードウェア保護の実現を目的として、軽量ハイパーバイザにより不揮発性メモリ領域への書き込みを遮断する手法を提案した。実装には BitVisor を用いて、コンセプト実証として Intel I218-V に搭載されている EEPROM への書き込みアクセスを遮断する実装について述べた。Netperf を用いたベンチマークテストにより、EEPROM への書き込みアクセスを遮断するための機構によるオーバーヘッドは非常に低く抑えられることを確認した。今後の課題としては、様々な種類のデバイスに搭載されている不揮発性メモリへの書き込みアクセスを防止する機構の実装や、オーバーヘッドを更に抑えるための最適化、より詳細な性能分析などが挙げられる。

参考文献

[1] Omote, Y., Shinagawa, T. and Kato, K.: Improving Agility and Elasticity in Bare-metal Clouds, *In Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 145–159 (online), DOI: 10.1145/2694344.2694349 (2015).
[2] Cooper, D., Polk, W., Regenscheid, A. and Souppaya, M.: BIOS Protection Guidelines, NIST Special Publication 800-147 (2011).
[3] Wojtczuk, R. and Tereshkin, A.: Attacking Intel BIOS, *BlackHat, Las Vegas, USA* (2009).
[4] Li, Y., McCune, J. M. and Perrig, A.: VIPER: Verifying the Integrity of PERipherals' Firmware, *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 3–16 (online), DOI: 10.1145/2046707.2046711 (2011).
[5] Shinagawa, T., Eiraku, H., Tanimoto, K., Omote, K., Hasegawa, S., Horie, T., Hirano, M., Kourai, K., Oyama, Y., Kawai, E., Kono, K., Chiba, S., Shinjo, Y. and Kato, K.: BitVisor: A Thin Hypervisor for Enforcing I/O Device Security, *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pp. 121–130 (online), DOI: 10.1145/1508293.1508311 (2009).
[6] IBM SoftLayer: `http://www.softlayer.com`.
[7] Internap: `http://www.internap.com`.
[8] Rackspace: `http://www.rackspace.com/`.
[9] Red Hat Bugzilla – Bug 459202 EEPROM/NVM of the e1000e becomes corrupted, `https://bugzilla.redhat.com/show_bug.cgi?id=459202`.
[10] Serious e1000e Driver Issue in SLE 11 Beta 1 and openSUSE 11.1 Beta 1, `https://news.opensuse.org/2008/09/22/serious-e1000e-driver-issue-in-sle-11-beta-1-and-opensuse-111-beta-1/`.
[11] Status of the e1000e Issue, `https://news.opensuse.org/2008/10/03/status-of-the-e1000e-issue/`.
[12] Delugrè, G.: How to develop a rootkit for Broadcom NetExtreme network cards, *Recon* (2011).
[13] Intel Corporation: Hacking Team's "Bad BIOS": A Commercial Rootkit for UEFI Firmware?, Technical report (2015).
[14] DEITYBOUNCE: `https://www.eff.org/files/2014/01/06/20131230-appelbaum-nsa_ant_catalog.pdf`.
[15] Comment on Der Spiegel Article Regarding NSA TAO Organization: http://en.community.dell.com/dell-blogs/direct2dell/b/direct2dell/archive/2013/12/30/comment-on-der-spiegel-article-regarding-nsa-tao-organization.
[16] Chen, P. M. and Noble, B. D.: When Virtual Is Better Than Real, *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, HOTOS '01, Washington, DC, USA, IEEE Computer Society, pp. 133– (online), available from ⟨http://dl.acm.org/citation.cfm?id=874075.876409⟩ (2001).

[17] Kourai, K. and Chiba, S.: HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection, *Proceedings of the 1st ACM/USENIX International Conference on Virtual Execution Environments*, VEE '05, New York, NY, USA, ACM, pp. 197–207 (online), DOI: 10.1145/1064979.1065006 (2005).

[18] Asrigo, K., Litty, L. and Lie, D.: Using VMM-based Sensors to Monitor Honeypots, *Proceedings of the 2Nd International Conference on Virtual Execution Environments*, VEE '06, New York, NY, USA, ACM, pp. 13–23 (online), DOI: 10.1145/1134760.1134765 (2006).

[19] Yang, J. and Shin, K. G.: Using Hypervisor to Provide Data Secrecy for User Applications on a Per-page Basis, *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pp. 71–80 (online), DOI: 10.1145/1346256.1346267 (2008).

[20] Jones, S. T., Arpaci-Dusseau, A. C. and Arpaci-Dusseau, R. H.: VMM-based Hidden Process Detection and Identification Using Lycosid, *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pp. 91–100 (online), DOI: 10.1145/1346256.1346269 (2008).

[21] Chubachi, Y., Shinagawa, T. and Kato, K.: Hypervisor-based Prevention of Persistent Rootkits, *Proceedings of the 2010 ACM Symposium on Applied Computing*, pp. 214–220 (online), DOI: 10.1145/1774088.1774131 (2010).

[22] Murray, D. G., Milos, G. and Hand, S.: Improving Xen Security Through Disaggregation, *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pp. 151–160 (online), DOI: 10.1145/1346256.1346278 (2008).

[23] Intel: Intel I/O Controller Hub 8 (Intel ICH8) Family Datasheet, `http://www.intel.co.jp/content/www/jp/ja/io/intel-io-controller-hub-8-datasheet.html` (2014).

[24] ethtool utility for controlling network drivers and hardware: `https://www.kernel.org/pub/ware/network/ethtool/`.

[25] The Netperf Homepage: `http://www.netperf.org/netperf`.

[26] Duflot, L., Perez, Y.-A. and Morin, B.: What if You Can't Trust Your Network Card?, *Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection*, RAID'11, Berlin, Heidelberg, Springer-Verlag, pp. 378–397 (2011).

[27] Yanlin, L., Jonathan, M. M. and Adrian, P.: VIPER: verifying the integrity of PERipherals' firmware, *In Proceedings of the 18th Proceedings of the 18th ACM conference on Computer and communications security*, pp. 3–16 (online), DOI: 10.1145/2046707.2046711 (2011).

[28] Fengwei, Z., Haining, W., Kevin, L. and Stavrou, A.: A Framework to Secure Peripherals at Runtime, *In Proceedings of the 19th European Symposium on Research in Computer Security*, Springer International Publishing, pp. 219–238 (2014).