ハードウェアによるメモリ監視を強制するためのIOMMU保護機構

荻野　堯[†]　味曽野　雅史[†]　品川　高廣[†]

1. はじめに

システムの完全性の検証や，メモリフォレンジックのためにはメモリ内容の取得や監視が必要となる．ハードウェアによるメモリ取得は，対象となるシステムに非侵襲でかつ高速にメモリ内容を取得することが可能であり，複数の研究でその手法が用いられている[1)~5)]．

一方で，最近はIOMMUを搭載したマシンが一般に広く普及している．IOMMUはPCIデバイスがDMAに利用するアドレスを変換する機構である．IOMMUにより非連続な領域に対するDMAや，DMA可能な範囲を制限することが可能となる．後者はデバイスドライバのバグの影響を最小限に抑えたり，悪意のあるデバイスからメモリを保護するために利用される．

IOMMUを有するマシンでハードウェアを利用したメモリ監視機構を利用する場合，もし攻撃者にマシンの制御が奪われると，攻撃者はIOMMU設定を書き換えることで，その機能を無効化する可能性がある．先行研究[1)~5)]はいずれもIOMMUが存在しない，あるいは無効化された環境を想定しているが，これでは正規のシステム利用者がIOMMUを利用できないことになる．

本研究ではIOMMUを有する環境下でも，ハードウェアによるメモリ監視機構を常に有効化するための手法を提案する．提案手法ではハイパーバイザを用いてゲストのIOMMU設定のシャドウィングを実施し，メモリ監視デバイスに対するIOMMU設定のパススルーを強制する．一方で，それ以外デバイスに対するIOMMUのマッピング設定は自由に変更を許可する．また，ハイパーバイザはIOMMU設定以外のメモリアクセスやデバイスアクセスは全てゲストにパススルーする．これにより，システムによる一般のIOMMU利用を許容しつつ，ハードウェアによるメモリ監視機能を常に有効にする．

2. IOMMU

IOMMUはデバイスがおこなうDMAのアドレスに関するメモリ管理ユニットであり，PCIの仕様としてはAddress Translation Service (ATS)という名称で定義されている．IOMMUの実装として，Intel VT-dやAMD IOMMUがある．

IOMMUの主な利用用途の一つは，仮想マシンに対してデバイスをパススルーする際にDMAの範囲をゲストメモリの範囲に制限することである．また，仮想マシンを利用しない場合でも，非連続領域に対するDMAやデバイスドライバのバグなどにより予期しない範囲にDMAされることを防ぐためにもIOMMUは利用される．

IOMMUのマッピング設定はメモリ上に配置されている．またIOMMUは各PCIデバイスごとにマッピングの設定をすることが可能である．

3. 脅威モデル

本研究では攻撃者がIOMMUを有するシステムを乗っ取り，OS上で任意のコード実行及びメモリ書き換えが可能な状況を想定する．一方で，ハイパーバイザに対する攻撃や，物理的なシステムへのアクセスは想定しない．攻撃者はメモリ上にあるIOMMU設定を変更することで，デバイスのDMAアクセスを無効化したり，あるいはDMAを特定のアドレスにリダイレクトすることを試みる可能性がある．

4. 提案手法

IOMMUがある環境下で攻撃者にマシンの制御を奪われた場合でも，ハードウェアによるメモリ監視機能（メモリ監視デバイス）を常に有効にするための方法を提案する．想定として，メモリ監視デバイスはシステム稼働時に最初からシステムに接続されているとする．

提案手法では，ハイパーバイザを用いてIOMMUの設定のシャドウィングをおこない，メモリ監視デバイスに対して常にメモリ全体にアクセス可能になるようなマッピングを設定する．メモリ監視デバイス以外のデバイスに対するIOMMU設定は，ゲストが設定したものをそのまま利用する．また，ハイパーバイザはIOMMU設定のシャドウィング以外のメモリアクセスやデバイスアクセスは全てパススルーとして扱う．これにより提案手法のオーバヘッドを抑え，また提案手法のTCBを小さくする．

メモリ監視デバイスに対するIOMMU設定は，パ

† 東京大学
The University of Tokyo

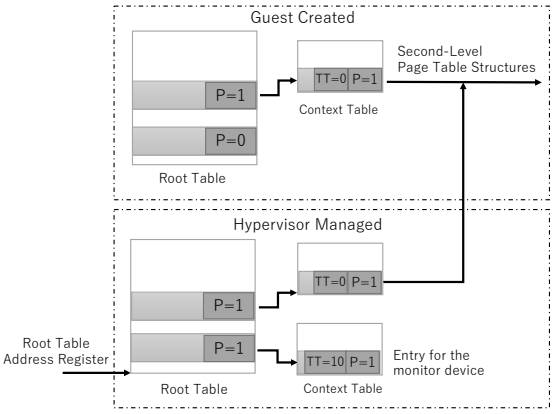

図 1 提案手法による IOMMU 設定のシャドウィング (VT-d)

スルー設定（IOMMU によるアドレス変換をおこなわない設定）が利用可能な場合はそれを，そうでない場合は DMA アドレスを恒等写像するマッピングテーブルを作成して利用する．

5. 実　　装

今回 Intel の IOMMU (VT-d) を対象に実装をおこなっている．図 5 に Intel VT-d における IOMMU 設定のシャドウィングの概要を示す．Intel VT-d では，Root table address レジスタにメモリ上にある IOMMU 設定のディスクリプタアドレスを設定する．IOMMU 設定は Root table, Context table 及び Second-level page table structures より構成される．PCI デバイスの bus, device, function 番号により Root table および Context table のエントリが一意に定まり，Second-level page table structures でそのデバイスに対する DMA のアドレス変換が規定される．

ここで，VT-d では Root table, Context table のエントリの Present bit (P) が 1 かつ Context table のエントリの Translation Type (TT) が 10 の場合，パススルーモードとなり，その設定が適用されたデバイスが利用するアドレスは IOMMU を介さずにそのまま DMA に利用される．

実装では EPT を用いて VT-d レジスタへのアクセスをトラップし，Root table 及び Context table のシャドウィングを実施する．この際にメモリ監視デバイスにはパススルーモードを設定する．それ以外の設定はゲストが作成したものをそのまま利用する．Second-level page table structures に関してはシャドウィングは実施しない．シャドウイングした IOMMU 設定は EPT によって保護する．これにより，ゲストの IOMMU 設定によらず，常に対象デバイスの IOMMU 設定がパススルーモードとなり，メモリ監視デバイスがメモリ内容を取得できることになる．

6. 関 連 研 究

ハードウェアによるメモリ取得を利用した研究として[1)~5)] などがある．これらの研究はいずれも PCI 接続のデバイスから DMA を利用してメモリ内容を取得をおこなうが，IOMMU の存在を想定しておらず，IOMMU の設定次第ではメモリ内容が取得できない，あるいは誤ったメモリ内容を取得する恐れがある．

[6)] は Linux は PCI の Address Translation Cache (ATC) に対応しているため，デバイスが ATC のケーバビリティ情報を OS に通知した上で，tranlated request として PCI の TLP を送信すると IOMMU 設定をバイパスできることを示したが，これも IOMMU の設定次第では無効化される恐れがある．

提案手法は IOMMU 設定をハイパーバイザで保護することにより，メモリ監視デバイスの IOMMU のパススルー設定を強制し，常に監視デバイスが動作するようにする．

7. まとめと今後の予定

本稿ではハードウェアによるメモリ監視を強制するための IOMMU 保護機構を提案した．現在 Intel の VT-d 向けに，軽量ハイパーバイザである BitVisor[7)] を用いて提案手法の実装を進めている．今後は DMA でメモリ内容の取得をおこなう PCILeech[8)] を利用して提案手法の有効性を検証する予定である．

参 考 文 献

1) N. L. Petroni, Jr., et al. Copilot - a Coprocessor-based Kernel Runtime Integrity Monitor. SSYM'04.
2) H. Moon, et al. Vigilare: Toward Snoop-based Kernel Integrity Monitor. CCS'12.
3) KI-Mon: A Hardware-assisted Event-triggered Monitoring Platform for Mutable Kernel Object. SEC'13.
4) L. Koromilas, et al. GRIM: Leveraging GPUs for Kernel Integrity Monitoring. RAID'16.
5) C. Spensky, et al. LO-PHI: Low-Observable Physical Host Instrumentation for Malware Analysis.
6) A. Atamli, et al. IO-Trust: An Out-of-band Trusted Memory Acquisition for Intrusion Detection and Forensics Investigations in Cloud IOMMU Based Systems. ARES'19.
7) T.Shinagawa, et al. BitVisor: A Thin Hypervisor for Enforcing I/O Device Security. VEE'09.
8) Ulf Frisk. Specifications - NVM Expres. `https://github.com/ufrisk/pcileech` (Visited on 2019-11-22).