Vol.2021-DPS-186 No.46
Vol.2021-CSEC-92 No.46
2021/3/16

準パススルー型ハイパーバイザによるメモリデータ収集機能の性能改善と評価

大森貴通 [1]　水野広基 [2]　牧原京佑 [2]　平野学 [1,2]　小林良太郎 [3]

概要：企業や官公庁でのランサムウェアの被害が増加している．我々の先行研究ではランサムウェア実行時のストレージ装置へのアクセスパターンを収集し，ランサムウェアを検知する機械学習モデルを訓練し，その検知性能を評価してきた．先行研究のシステムではランサムウェアをある程度の精度で検知できていたが，ストレージ装置へのアクセスパターンがランサムウェアと類似している無害なプログラムをランサムウェアと誤検知してしまう問題があった．そこで本研究ではランサムウェアの振る舞いをモデル化するために，先行研究で扱ったストレージ装置へのアクセスパターンに加えて，メインメモリから得られるプログラムの振る舞いに関する特徴量を収集し，それらの両方を用いることでランサムウェアの検知性能を向上させることを目的とする．先行研究では準パススルー型ハイパーバイザを用いてランサムウェアのストレージ装置に対するアクセス履歴を収集するシステムを実装，利用してきた．本研究ではこの先行研究のシステムを拡張し，メインメモリへのアクセス履歴を収集する機能を追加する．本稿では以下の 2 つの試作をおこなった結果を報告する．まず，試作（1）では物理メモリマップの情報を用いてゲスト OS のメモリ領域をバッファにコピーし，監視サーバへ転送することを繰り返す．試作（2）では Intel 社製 CPU の仮想化技術のひとつである Extended Page Table を利用し，メモリにアクセスされたタイミングで物理アドレスとそのデータを収集する．本稿ではそれぞれの試作を性能評価した結果を報告する．最後にメモリ管理の仮想化を応用した最新の関連研究をいくつか示し，監視の頻度と性能を両立できるメモリ監視システムの設計について考察する．

キーワード：ハイパーバイザ，ランサムウェア，仮想化，メモリ管理，フォレンジック

1. はじめに

マルウェアつまり悪意のあるプログラムは，企業，官公庁，軍事，社会インフラ等の組織のシステムに侵入し，データを改竄，破壊，窃取，場合によっては長期にわたって諜報活動をするために悪用される．図 1 にドイツの IT セキュリティ研究機関 AVTEST が公開しているマルウェアの検体数の推移[1]を示す．このようにマルウェアの報告件数は増加を続けているが，現在の多くのアンチウィルス製品が採用する「シグネチャ型」の検知手法では，マルウェア解析の専門家が手作業によって検体を解析してシグネチャを作成するため，ある程度の時間とコストを要する．ここでシグネチャとは，特定のマルウェア検体に共通する一続きのバイト列のことである．最近は一部のコードだけを変更した亜種が増加している問題に対応するため，機械学習によってシグネチャを自動生成したり，それを支援する研究も行われている．しかしながらシグネチャを作成することができるのは既知のマルウェアだけであり，「シグネチャ型」には根本的に未知のマルウェアを検知できないという欠点がある．しかしながら「シグネチャ型」は誤検知が少なく実用化に適した手法として定着している．

近年はシグネチャを用いずに，マルウェアの外的な振る舞いを用いて検知する「振る舞い型」のシステムも提案されつつある．振る舞いのモデル化には，特徴量としてネットワーク通信や，マルウェアが用いるシステムコールを利用するのが一般的である．我々の先行研究[2]では準パススルー型ハイパーバイザである BitVisor [3]を使用して，ストレージ装置，すなわち Hard Disk Drive (HDD) や Solid State Drive (SSD) へのアクセスパターンを特徴量としてランサムウェアを検知するシステムを試作，評価してきた．ランサムウェアとは，ユーザのファイルを暗号化して身代金を要求するマルウェアの一種である．ランサムウェアの目的は侵入したシステムのストレージ装置にあるファイルを短時間にできるだけ多く暗号化することであるため，ストレージ装置のアクセスパターンを隠すことが難しく，ランサムウェアの振る舞いのモデル化に有効であった．先行研究[2]ではストレージアクセスパターンで訓練した機械学習モデルで多くのランサムウェアを悪性と検知できたものの，一部のランサムウェアの検知性能が低く，さらにランサムウェアと似た振る舞いを持つ無害なソフトウェア（暗号化や圧縮プログラム）をランサムウェアと誤検知してしまう課題もあった，さらに提案システムのままではストレージ

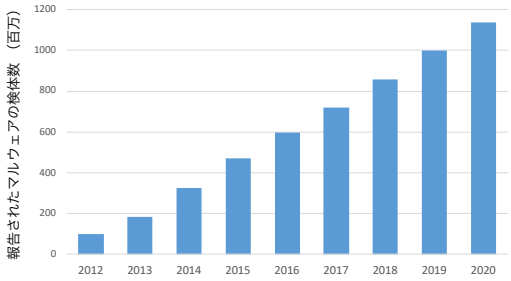

図 1　AVTEST が公開したマルウェアの検体数

1 豊田工業高等専門学校 専攻科 情報科学専攻
2 豊田工業高等専門学校 情報工学科
3 工学院大学 情報学部

ⓒ2021 Information Processing Society of Japan

Vol.2021-DPS-186 No.46
Vol.2021-CSEC-92 No.46
2021/3/16

装置をまったく介さない新しいマルウェア（ファイルレスマルウェア等）に適用できない課題もあった．

そこで，本研究では先行研究[2]で採用したストレージ装置のアクセスパターンに加えて，ランサムウェアの振る舞いの新しい特徴量としてメインメモリへのアクセスパターンを検討する．特に本稿では軽量の準パススルー型ハイパーバイザをメモリ監視に利用する方法を説明し，二通りの試作をおこなった結果を報告する．試作したシステムは（1）メモリマップの情報をもとに特定の範囲の物理メモリ領域を監視サーバへ転送する方式，（2）Intel 社製 CPU の仮想化技術のひとつ Extended Page Table を利用してアクセスされたタイミングで物理アドレスとデータを収集して監視サーバへ転送する方式である．最後にメモリ仮想化を応用した最新の関連研究を示しながら，性能と監視を両立するためのメモリ監視システムの実現方法について考察する．

2. 背景

2.1 準パススルー型ハイパーバイザ

本研究で利用する BitVisor [3] は Intel 社の x86 CPU プラットフォーム向けの仮想化技術 VT-x を用いて実装された軽量のハイパーバイザである．BitVisor が採用する準パススルー方式は，監視対象のハードウェア資源だけを仮想化し，それ以外のハードウェア資源はゲスト OS へそのまま通過させることで，仮想化によるゲスト OS の性能の低下を最小化する．本研究では BitVisor をランサムウェアの振る舞いを収取するために利用する．他の汎用的なハイパーバイザと比べて仮想化なしの元の環境に近い状態でランサムウェアの振る舞いを収集できると考えられる．

メインメモリの振る舞いを取得するという観点からいうと，BitVisor はゲスト物理アドレスとホスト物理アドレスを基本的に変換せずに，そのまま一対一対応させる．よって，仮想化なしと同じ状態でメモリへのアクセスパターンを収集でき，メモリ監視の実装に適している．これは BitVisor が仮想化層で入出力の暗号化やアクセス制御などのセキュリティ機能を強制することを目的として設計されていて，単一のゲスト OS しかサポートしていないことによる．メモリ監視に関して BitVisor は Shadow Page Table と Extended Page Table (EPT) の両方をサポートしているが，本研究では後者を利用するため，その詳細を次に説明する．

2.2 Extended Page Table によるメモリ仮想化

Intel 社が提供する Software Developer Manuals 28 章には VMX (Virtual Machine eXtension) 命令に含まれる機能のひとつとして Extended Page Table (EPT) が示されている[4]．EPT の目的は VMX non-root operation 時にゲスト物理アドレスをホスト物理アドレスへ変換することである．図 2 にゲスト OS のプロセスまたはカーネルが自身のページテーブルでゲスト仮想アドレスをゲスト物理アドレスへ変換し，ハイパーバイザである BitVisor が EPT を用いてゲスト物理アドレスをホスト物理アドレスへ変換する流れを示す．EPT のデータ構造は，ゲスト OS が仮想アドレス（すなわちリニアアドレス）を解決するのに用いる Page Table に類似している．EPT ではページ単位で権限（Privilege）を指定でき，その権限に違反するアクセス（EPT Violation と呼ばれる）が起きると，VM exit によってゲスト OS からハイパーバイザに処理が遷移する．ハイパーバイザを用いてメインメモリを監視するセキュリティ研究の多くは この EPT Violation を利用しており，本研究でも EPT をメモリアクセスの監視に利用する．なお，ゲスト物理アドレスのエントリがまだ EPT ページング構造体に存在していない時のアクセスでも EPT Violation が発生する．

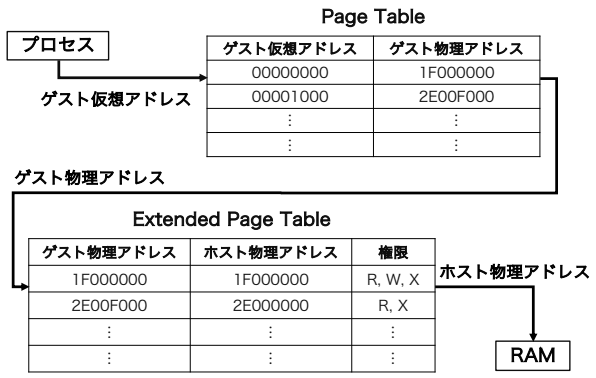

図 2　アドレス変換の概略

EPT でゲスト物理アドレスからホスト物理アドレスへ変換する際には 4 種類の EPT ページング構造体（それぞれ 512 個ずつ）が使われ，それら 4 つを ”ページウォーク” することでアドレス変換を実現する．特に最初の EPT ページング構造体である PML4E へのポインタは Extended Page Table Pointer (EPTP) と呼ばれるデータ構造に含まれている．各 EPT ページング構造体は PML4E，PDPTE，PDE，PTE と呼ばれ，それぞれ 512 GiB，1 GiB，2 MiB，4 KiB の粒度でアドレス変換を処理する．EPT では EPT ページング構造体ごとの粒度でアクセス権限の設定をおこなうことができるが，本稿の試作では最も粒度の細かい 4 KiB 単位でアクセス権限を設定して，意図的に EPT Violation を発生させることでメモリアクセスを監視する．

2.3 EPT Violation と Translation Lookaside Buffer

EPT を利用したメモリアクセス監視システムを設計する際にはキャッシュの存在を考慮しなければならない．CPU に内蔵されているキャッシュには L1 キャッシュ，L2 キャッシュ，L3 キャッシュに加えて，アドレス変換をキャッシュする Translation Lookaside Buffer (TLB) があり，TLB にはデータ用 TLB と命令用 TLB がある．EPT Violation は EPT にエントリがない場合や権限がない場合に発生するが，アドレス変換が TLB にキャッシュされてしまうと EPT Violation が発生しなくなり，メモリアクセスを監視できなくなる．CPU に内蔵された TLB はソフトウェアの仲介を

ⓒ2021 Information Processing Society of Japan

Vol.2021-DPS-186 No.46
Vol.2021-CSEC-92 No.46
2021/3/16

必要とせず，過去のアドレス変換を自動的にキャッシュしていく．TLB のキャッシュがクリアされるのは以下の場合である[4]：（1）INVLPG や INVPCID のような TLB をクリアする命令が発行された時，（2）EPT Violation が発生した時，EPTP に関連付けられたゲストとホストのマッピングをクリアする，（3）INVVPID 命令を発行した時，VPID ごとの粒度で TLB をクリアする，（4）INVEPT 命令を発行した時，EPTP ごとの粒度で TLB をクリアする．本研究では，深井らがベアメタルクラウドのライブマイグレーションを BitVisor で実装した際に用いた TLB shootdown[6]を採用し，すべてのコアの TLB をクリアするようにした．

3. メモリデータ収集機能の試作

本稿ではメモリマップを用いて特定領域をバッファへコピーして監視サーバへ転送する方式と，EPT Violation によってアクセスしたタイミングでアクセス履歴を収集する方式の２つを試作した結果を報告する．まず，それらの試作のベースとなった先行研究の監視システムから紹介する．

3.1 先行研究の監視システムとメモリ監視への拡張

図 3 に先行研究の監視システム[7]にメモリ監視機能を追加した構成図を示す．先行研究の監視システムはストレージ装置へのアクセスパターンだけを収集していたが，今回はさらにメモリデータの収集機能を追加実装した．準パススルー型ハイパーバイザ BitVisor は EPT によるメモリ仮想化をサポートしているため，その機能を利用してメモリのアクセスパターンを収集する．また，BitVisor には TCP/IP スタックが実装されているため，それを用いて NTP プロトコルを実装し，収集した個々のアクセスパターンのタイムスタンプを記録できるようにした．メモリのアクセスパターンは一旦データ収集用バッファにコピーされる．それらのデータは BitVisor に実装されている UDP スタックの機能を利用して一定間隔ごとに監視サーバへ転送する．アクセスパターンの転送には 10 GbE NIC を利用するため理論上の上限値は 1.25 GB/s である．本研究ではこの上限値を考慮したシステム設計をおこなう．監視の開始と停止はゲスト OS から VMCALL を用いる設計とする．

3.2 試作（1）物理メモリマップを用いた一括転送

まずは EPT を用いずに，物理メモリのうちゲスト OS が利用している領域だけを BitVisor のデータ収集用バッファに一旦コピーし，監視サーバへ転送する方式を試作した．物理メモリは時々刻々と変化するため，一旦バッファにコピーした後にゲスト OS へ負荷をかけずに少しずつ監視サーバへ転送する．大山らは BitVisor を用いて OS の状態を復元する機構を提案している[8]．これは任意の時点のゲスト OS の状態を保存しておき，後でその状態に復元するものである．大山らの実装ではメモリの保存と復元に物理メモリマップを利用し，usable とマークされた領域だけを取得することでシステムを実装した．本稿の試作でも同様の

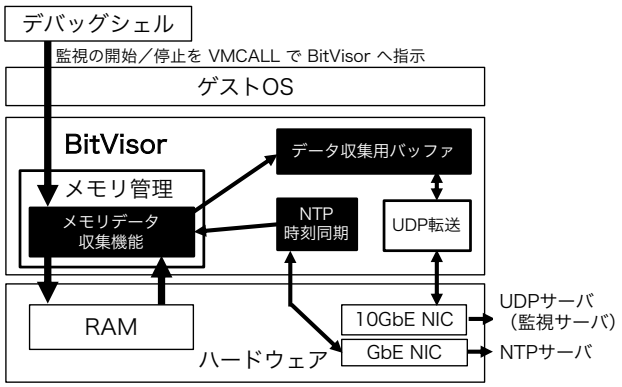

図 3　先行研究システムとメモリ監視への拡張

```
e820: BIOS-provided physical RAM map:
BIOS-e820: [mem 0x0000000000000000-0x000000000000fff] ACPI NVS
BIOS-e820: [mem 0x0000000000001000-0x000000000009efff] usable
BIOS-e820: [mem 0x000000000009f000-0x000000000009ffff] unusable
BIOS-e820: [mem 0x00000000000c0000-0x00000000000fffff] reserved
BIOS-e820: [mem 0x0000000000100000-0x000000000ef24fff] usable
BIOS-e820: [mem 0x000000000ef25000-0x000000000ef2dfff] reserved
BIOS-e820: [mem 0x000000000ef2e000-0x000000000ef3cfff] usable
BIOS-e820: [mem 0x000000000ef3d000-0x000000000ef42fff] reserved
```

図 4　e820 命令で取得した物理メモリマップの例

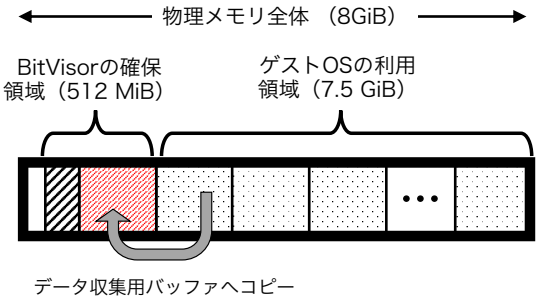

図 5　試作（1）の物理メモリ取得方式

方法で実装をおこなった．これによって BIOS が起動時に割り当てた物理メモリ領域，つまり OS が使ってはいけない領域にアクセスしてシステムを停止させることなく，ゲスト OS のメモリ空間だけをコピーできるようになった．図 3 に e820 命令を用いて取得した物理メモリマップの例を示す．インテルアーキテクチャ（IA）ではリアルモードで e820 命令を発行することでメモリマップを取得することができる．本試作でも大山らの研究[8]と同様に BitVisor の物理メモリマップ取得機能を利用した．

図 5 に試作（1）の物理メモリ取得方式を示す．この例では物理メモリ 8GiB のマシンで BitVisor に 512 MiB を割り当て，残りの約 7.5 GiB をゲスト OS に割り当てている．BitVisor に割り当てた 512 MiB のうち一部をデータ収集用バッファとし，ゲスト OS が利用する領域のうち usable の部分だけを一定間隔でデータ収集用バッファにコピーする．その後，ゲスト OS の性能が低下しないように，BitVisor が別スレッドでデータ収集用バッファのデータを監視サーバへ UDP 転送することを繰り返す設計とした．

ⓒ2021 Information Processing Society of Japan

Vol.2021-DPS-186 No.46
Vol.2021-CSEC-92 No.46
2021/3/16

3.3 試作（２）EPT を用いたアクセスパターンの収集

EPT を用いたアクセスパターンの収集では以下の２通りの試作をおこなった．まず，試作（２ａ）ではアクセスされた物理アドレス，アクセス種別（読み込みまたは書き込み)，タイムスタンプを監視サーバへ送信する設計とした．図 6 に試作（２ａ）のフローチャートを示す．BitVisor のメインループが n 回繰り返されるごとに EPT または TLB，またはその両方をクリアすることで，EPT Violation を発生させ，そのタイミングでメモリアクセスを収集する．TLB と EPT のクリアを頻繁に行うとゲスト OS の性能に悪影響を及ぼすため頻度を n の値で調整できるようにした．

EPT に存在しないページエントリに初めてアクセスする際には EPT Violation が発生するが，一旦エントリが EPT に存在するようになると，次に EPT がクリアされるまでは EPT Violation が発生しなくなる．つまり，試作（２ａ）では前のクリアと次のクリアの間に同じアドレスへのアクセスが 2 回以上発生した場合，2 回目以降のアクセスを捕捉できない．また，CPU キャッシュにアクセスされたページがある場合のメモリアクセスも捕捉できない．しかしながら，ランサムウェアのように大量のデータを読み書きする場合には CPU キャッシュの影響をある程度無視しても大まかな傾向を得られると考えている．BitVisor ではゲスト物理アドレスとホスト物理アドレスは常に同じとなるため，EPT がクリアされても，その後 EPT エントリの追加と削除が発生するだけであり，それ以外の処理は行われない．

図 7 に試作（２ａ）を用いて実際に実行マシンから監視サーバへ送られたメモリアクセス履歴の一例を示す．捕捉したメモリアクセス 1 回ごとに 32 バイトのデータを送信する．先行研究[2]では監視サーバへ送られたストレージ装置へのアクセスパターンを特徴量に変換してランサムウェアを検知するための機械学習モデルを構築した．本研究で収集するメモリアクセスパターンも将来的にランサムウェアを検知するための機械学習モデルとして利用することを想定している．

試作（２ａ）では読み込みと書き込みのアドレスだけを収集したが，次の試作（２ｂ）では書き込まれたデータとその物理アドレスを収集する設計とした．試作（２ｂ）は試作（２ａ）と異なり，開始時にだけ EPT と TLB をクリアする．その後は EPT Violation が発生するたびにアクセスされたページの EPT エントリの Write 権限を削除することで，継続的に書き込み時の EPT Violation を発生させる設計とした．監視の頻度を調整するため BitVisor のメインループが m 回繰り返されるごとに，書き込まれたデータを監視サーバへ転送する設計とした．ここで書き込みの EPT Violation が発生したタイミングはアドレス変換の段階であり，物理メモリへの書き込みが完了していないことに注意が必要である．試作では過去 k 件の書き込み要求の物理アドレスをアクセス履歴配列に記憶しておき，一定時間が

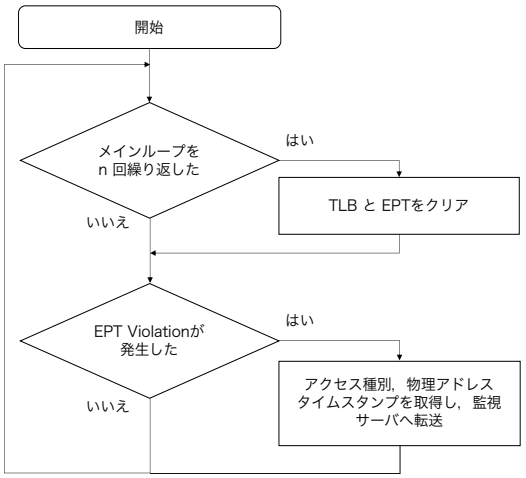

図 6　試作（２ａ）のフローチャート

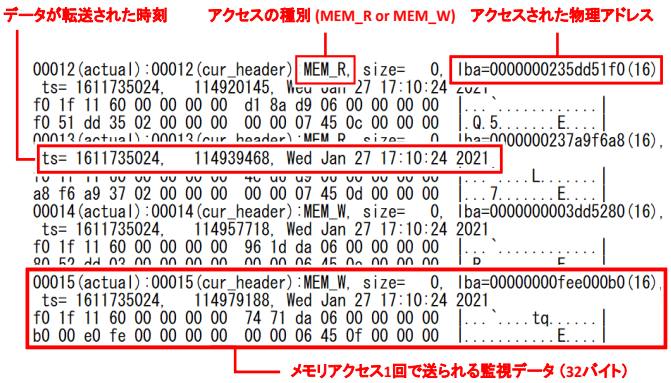

図 7　監視サーバで受信されたメモリアクセス履歴

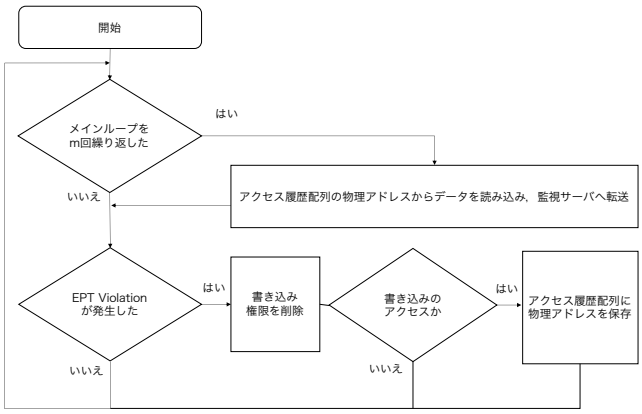

図 8　試作（２ｂ）のフローチャート

経過してからその物理アドレスのデータをデータ収集用バッファにコピーして転送する設計とした（現在の試作では k＝7 としている）．さらに本来は CPU キャッシュを考慮しないといけないが，現時点の試作では CPU キャッシュからメモリに書き戻されたデータを取得できているかまでは検証できていない．しかしながら試作（２ａ）と同様，ランサムウェアのように大量のデータを読み書きする場合には CPU キャッシュの影響をある程度無視しても，ランサムウェア特有のアクセスパターンを収集できると考えている．

ⓒ2021 Information Processing Society of Japan

Vol.2021-DPS-186 No.46
Vol.2021-CSEC-92 No.46
2021/3/16

表 1　性能評価に用いたマシンの仕様

ハードウェア	規格
CPU	Intel Celeron G3920 2.9GHz (2 Core)
	L1 データキャッシュ　2 x 32 KB
	L1 命令キャッシュ　2 x 32 KB
	L2 キャッシュ　2 x 256 KB
	L3 キャッシュ　2 MB（2 コアで共有）
	データ TLB　1 GB, 4-way, 4 エントリ
	4 KB, 4-way, 64 エントリ
	命令 TLB　4KB, 8-way, 128 エントリ
	最大メモリ帯域幅　34.1 GB/s
RAM	DDR4-2133，8GB
マザーボード	ASRock H110M-HDV
ネットワーク	Intel X550-T1 (10GbE), Intel PRO/1000 PT (GbE)

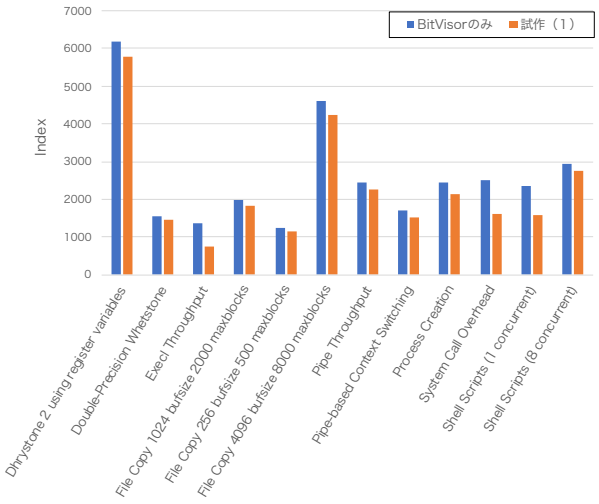

図 9　試作（1）　UnixBench の実行結果

4. 性能評価

試作したプログラムの性能を評価した．実験に用いたマシンの仕様を表 1 に示す. BitVisor は 2018 年 12 月 4 日にリポジトリ[5]から取得したソースコードを元に，試作（1），試作（2 a），試作（2 b）を個別に実装した．実行マシンと監視マシンは 10 GbE ネットワークで接続し，メモリアクセス履歴を監視マシンへ転送できるようにした．

4.1 試作（1）物理メモリマップを用いた一括転送

BitVisor に 512 MiB を割り当て，そのうちの 192 MiB（このうちデータ領域は 96MiB）をデータ収集用バッファとして割り当てた．メモリマップで得た領域 42 個のうち 11 個がゲスト OS が使える usable の領域であり，その合計は 6.54 GiB であった．ゲスト OS として CentOS 7 を動作させ監視サーバへ 6.54 GiB の転送を繰り返し実行させている間に UnixBench [9]でゲスト OS の性能評価を行った．図 9 に示す結果は CPU コア 2 つでベンチマークを 10 回実行した時の平均値であり，Index 値が大きいほど性能が良いことを示している．Excel Throughput の項目で性能が 45% 低下し，全項目の平均では性能が 16% 低下した．

4.2 試作（2）EPT を用いたアクセスパターンの収集

試作（2 a）の BitVisor を用いて，CentOS 7 をゲスト OS として実行し，その上でメモリ帯域幅を計測するマイクロベンチマーク STREAM[10]を実行した．一定間隔で EPT だけをクリア，TLB だけをクリア，EPT と TLB の両方をクリアの 3 つの条件で実験した．図 10 に EPT だけをクリアした時の結果を，図 11 に TLB だけをクリアした時の結果を，図 12 に EPT と TLB を両方ともクリアした時の結果を示す．折れ線グラフは 1 秒あたりに監視できたページ数を示しており，値が大きいほうが望ましい．棒グラフは STREAM の処理時間を示しており，値が小さい時に性能が良いことを示している．

STREAM は各要素が 8 バイト，要素数が 10,000,000 個の 3 つの配列（配列ごとに 76.3 MiB のメモリを消費する）$A[i]$，$B[i]$，$C[i]$ を用いて，以下の 4 つのメモリ処理に要する時間を計測する．

- Copy 処理：$C[i] = A[i]$
- Scale 処理：$C[i] = \text{scalar} * A[i]$
- Add 処理：$C[i] = A[i] + B[i]$
- Triad 処理：$C[i] = A[i] + \text{scalar} * B[i]$

今回の実験結果すべてにおいて，3 つの配列を用いる Add 処理と Triad 処理は，2 つの配列しか使わない Copy 処理と Scale 処理よりも長い時間を要していた．EPT のクリアだけの場合（図 10）と TLB のクリアだけの場合（図 11）は処理時間が大幅に増加することはなかったが，EPT と TLB を両方クリアした場合（図 12）に n=100 で処理時間が大幅に増加した．BitVisor の EPT クリア（図 10）は内部で一部の TLB クリアが実行されているが，本実験の TLB クリア（図 11）は TLB shootdown [6]を実行するため，両方を実行することでより高い負荷がかかったと考えられる．

次に 1 秒間に監視できたページ数に関しては，図 10 に示すように EPT だけをクリアした場合に最も多くのページを監視でき，逆に図 11 に示すように TLB だけをクリアした場合には平均で 40 個/s 程度のページへのアクセスしか監視できなかった．表 1 に示したように今回使用した CPU に内蔵されている 4 KiB ページ用の TLB のエントリ数は 64 個または 128 個であったため想定内の結果といえる．一定間隔で TLB をクリアすることによって特に頻繁にアクセスされるページへのアクセスを収集できるかもしれないが，収集できるページ数は TLB のエントリ数に制限されることを確認できた．次に図 10 の折れ線グラフに注目すると n=150 あたりで傾向が変化していることがわかる．もし n が大きければ（監視の頻度が少なければ）EPT に入るアドレス変換のエントリ数が増え，次に EPT がクリアされるまでは，一旦エントリに格納されたアドレスの EPT Violation は発生しなくなる．今回の実験では $n < 150$ の時に，同じアドレスへ繰り返されるアクセスを補足した結果，監視できたページ数が増加したと推測される．

ⓒ2021 Information Processing Society of Japan

Vol.2021-DPS-186 No.46
Vol.2021-CSEC-92 No.46
2021/3/16

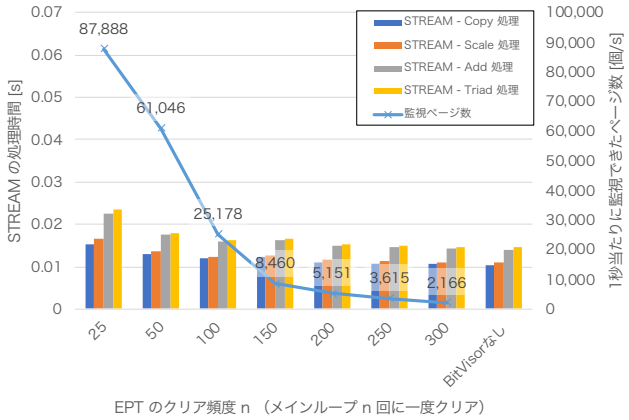

図 10　試作（２ａ）EPT のみクリアした時の実験結果

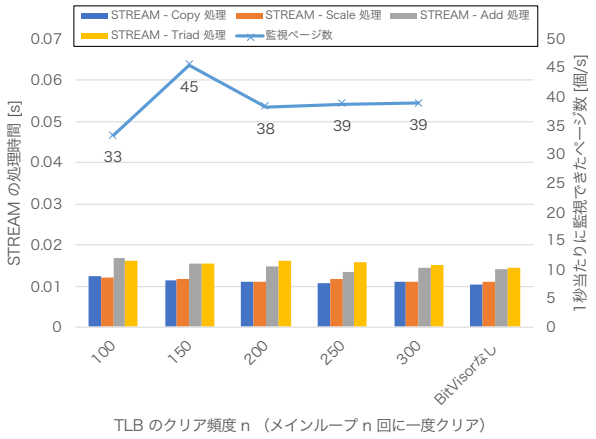

図 11　試作（２ａ）TLB のみクリアした時の実験結果

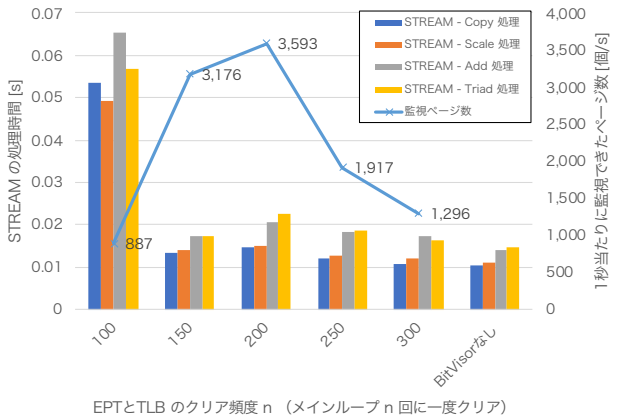

図 12　試作（２ａ）EPT と TLB を両方クリアした時の実験結果

しかしながら，この点については実際に取得したアドレスのヒストグラムを作成するなどして更に検証する必要がある．次に，図 12 に注目すると，TLB と EPT を両方クリアした場合，n<200 で監視できたページ数が減少傾向となり，n=100 でゲスト OS のメモリ処理性能が大幅に低下し，かつ監視できたページ数も急減した．

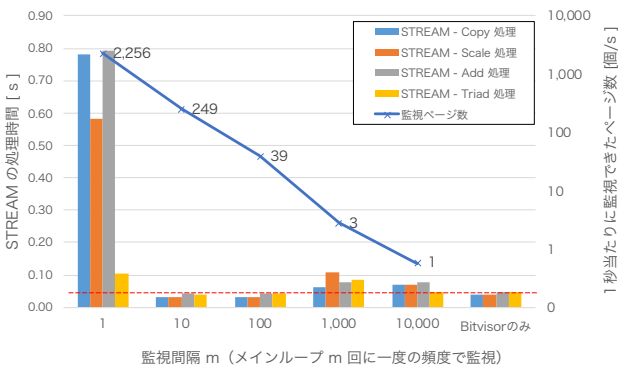

図 13　試作（２ｂ）EPT と TLB をクリアせず，すべての EPT Violation で Write 権限を削除した時の実験結果

最後に試作（２ｂ）の BitVisor を用いて，Windows 7 をゲスト OS として実行し，その上で STREAM を実行した時の結果を図 13 に示す．Windows 7 では STREAM の実行結果が正しく計算されないことがあったため，配列の要素数を 3 倍の 30,000,000 個（配列ごとに 686.6 MiB のメモリを消費）に，試行回数 NTIMES の値を 10 回から 100 回に変更して実験した．試作（２ｂ）では TLB と EPT をクリアせず，EPT Violation が発生した時に常に Write 権限を削除することで，継続的に EPT Violation を発生させ，書き込みのアクセスを収集する．このため TLB にキャッシュされたアドレス変換は原理的に捕捉できていないはずである．

図 13 に示す試作（２ｂ）の STREAM の処理時間は，図 10，図 11，図 12 で示した試作（２ａ）より長くなっている．たとえば，図 10 で n=300 の時に 4 つの処理時間の平均値が 0.013 s だったのに対し，図 13 では 10,000 回に 1 回（m=10,000）で監視したときは 0.066 s と実に 5 倍ほど処理時間を要した．これには以下の理由が考えられる．まず，試作（２ａ）では物理アドレスだけを収集していたのに対し，試作（２ｂ）は実際に書き込まれたデータを収集している．とくに現在の実装では全ての EPT Violation で VM exit が発生し，そこで物理アドレスをアクセス履歴配列へ保存する．その後，一定時間経過してからその物理アドレスからデータを取得して監視サーバへ転送する処理が追加されている．さらに，１回のアクセスで監視サーバへ送信するデータ量が試作（２ａ）では 32 バイトであったのに対し，試作（２ｂ）では 32 バイトと 4096 バイトの計 4,128 バイトに増加している．以上の追加処理が STREAM の結果（すなわちゲスト OS でのメモリ帯域幅）を悪化させたと考えられる．

EPT Violation の頻度に関しては，図 10 で EPT のクリア頻度が最も少ない n=300 のときに，捕捉できたアクセス数（=EPT Violation 数）は 2,166 個/s であったのに対して，図 13 で EPT をクリアしなかったときの EPT Violation 数（m=1）は 2,256 個/s であった．これら２つの値は近い

ⓒ2021 Information Processing Society of Japan

Vol.2021-DPS-186 No.46
Vol.2021-CSEC-92 No.46
2021/3/16

値であり，EPT によるアドレス変換が定常的にどの程度発生しているのかを示唆していると考えられる．

以上の結果からおおよその EPT と TLB の挙動を推定することが可能になったが，実験（２ａ）は CentOS 7 での結果，実験（２ｂ）は Windows 7 での結果であるため，今後は実験条件を揃えて検証する必要がある．また，試作（２ａ）と試作（２ｂ）では UnixBench のようなゲスト OS の性能を総合的に評価するベンチマークを実行していないため，ゲスト OS へ与える影響の追加検証が必要である．

5. 考察

前節までででハイパーバイザを用いたメモリアクセスを監視する機能の設計と試作を示し，性能評価の結果を示した．ここでは特にメモリ仮想化を用いた最新の研究成果を示しながら，本研究に関連する事項について考察する．深井らはベアメタルクラウド，すなわち仮想マシンではなく物理マシンを提供する Infrastructure as a Service (IaaS)，で可能となる新しい攻撃を示した．そして，それに対抗するための BitVisor をベースとした保護機構 BMCArmor[11]を提案し，BIOS ROM と Intel 社 NIC の EEPROM を保護できることを示した．BMCArmor は Memory-mapped IO (MMIO) 経由の書き込みを制限するため，予め該当する EPT エントリの書き込み権限を操作しておき EPT Violation を発生させる．深井らの研究では監視するメモリ領域を限定することで，VM exit による性能低下を最小化している．

一方，Hua らは EPT をカーネル用とユーザ用に分けることによって Meltdown 攻撃の原因となるサイドチャネルを効果的に除去した[12]．この研究の特徴は Intel 社製 CPU の VMFUNC 命令を利用している点にある．VMFUNC 命令はハイパーバイザの機能を VMX non-root operation から VM exit なしで実行できるものである[4]．この研究では VMFUNC 命令の代表的な用途である EPTP (EPT pointer) Switching を用いて VM exit せずに，カーネル用の EPT とユーザ空間用の EPT を切り替える提案を行った．このとき，EPTP だけの値が更新され，CR3 には影響を与えない．それぞれの EPT には独自の TLB が VPID によって割り当てられるため無駄なキャッシュの Invalidation が発生せず，高い性能を維持できる．我々の研究に EPT Switching を応用できればユーザ空間のメモリアクセス性能を悪化させずに，カーネル空間だけを効率的に監視できるかもしれない．

また，Mi らはクラウドコンピューティングを構成するハイパーバイザの安全性を高めるために Nested virtualization を導入し，信頼できないゲスト VM を追加の仮想化層で保護する Guardian VM を提案している[13]．この提案でもハイパーバイザを用いてメモリ保護をおこなう機構に共通の課題，すなわち VM exit が増えることによる性能低下を解決する工夫のひとつとして VMFUNC 命令による EPTP Switching を採用している．

6. まとめ

本稿ではランサムウェアの振る舞いをモデル化するために，先行研究[2]で用いたストレージ装置へのアクセスパターンに，新たにメインメモリへのアクセスパターンの特徴量を追加することを目的として，準パススルー型ハイパーバイザ BitVisor を用いた２つのメモリ収集方式の試作と評価をおこなった．一括転送は VM exit による性能低下があまりないが，EPT Violation によるメモリ監視は VM exit による性能の低下が不可避である．今後は両者を組み合わせることで，より効率的な新しいメモリ監視機構を提案，評価していく計画である．

謝辞 本稿の試作システムの実装にあたり，筑波大学学術情報メディアセンターの大山恵弘氏の実装[8]を参考にさせていただきました．ここに深く感謝の意を表します．本研究は JSPS 科研費 20K11825 の助成を受けたものです．

参考文献

[1] AVTEST: The Independent IT-Security Institute, https://www.av-test.org/en/statistics/malware/, 2021 年 2 月 11 日閲覧.
[2] Hirano, M. and Kobayashi, R. Machine Learning Based Ransomware Detection Using Storage Access Patterns Obtained from Live-forensic Hypervisor. In 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), IEEE, pp. 1-6, 2019.
[3] Shinagawa, T. , Eiraku, H. et al. “BitVisor: A Thin Hypervisor for Enforcing I/O Device Security” ACM VEE 2009, pp. 121-130, 2009.
[4] Intel Corporation, Intel® 64 and IA-32 architectures software developer’s manual. Volume 3: System Programming Guide, Nov. 2020.
[5] BitVisor, https://www.bitvisor.org, 2021 年 2 月 11 日閲覧.
[6] Fukai, T., Shinagawa, T., and Kato, K. Live migration in bare-metal clouds. IEEE Transactions on Cloud Computing, 2018.
[7] Hirano, M. et al. WaybackVisor: Hypervisor-based scalable live forensic architecture for timeline analysis. In International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, pp. 219-230, Springer, Cham, 2017.
[8] Oyama, Y., Kawasaki, Y., and Takahashi, K. Checkpointing an Operation System Using a Parapass-throgh Hypervisor, Journal of Information Processing, Vol. 23, pp. 132-141, 2015.
[9] UnixBench, https://github.com/kdlucas/byte-unixbench, 2021 年 2 月 11 日閲覧.
[10] John D. McCalpin, STREAM: Sustainable Memory Bandwidth in High Performance Computers, https://www.cs.virginia.edu/stream/, 2021 年 2 月 11 日閲覧.
[11] Fukai, T., Takekoshi, S., Azuma, K., Shinagawa, T. and Kato, K. BMCArmor: A Hardware Protection Scheme for Bare-Metal Clouds. In 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 322-330, 2017.
[12] Hua, Zhichao et al. EPTI: Efficient Defence against Meltdown Attack for Unpatched VMs. In 2018 USENIX Annual Technical Conference (USENIX ATC 18), pp. 255-266. 2018.
[13] Mi, Zeyu, Dingji Li, Haibo Chen, Binyu Zang, and Haibing Guan. (Mostly) Exitless VM Protection from Untrusted Hypervisor through Disaggregated Nested Virtualization. In 29th USENIX Security Symposium (USENIX Security 20), pp. 1695-1712. 2020.

ⓒ2021 Information Processing Society of Japan