Vol.2009-ARC-183 No.31
Vol.2009-OS-111 No.31
2009/4/24

# 仮想マシンモニタによるゲストOSのファイル保護

忠鉢洋輔[†1]　品川高廣[†2]　加藤和彦[†2]

近年，OSを改変しカーネル内で不正な動作をするプログラムを用いた攻撃が一般化している．このとき，従来のOSによるアクセス制御が無効にされOSが保護されず，被害の拡大に繋がる可能性がある．本研究では，OSが信頼できない状態でもファイルの保護を可能にするために，仮想マシンモニタ（VMM）を用いたゲストOSのファイルに対するアクセス制御手法を提案する．ファイルとストレージとの間の“semantic gap”に対処するために，予め安全な状態のゲストOS上でファイルと論理ブロックアドレス（LBA）との対応関係を作成し，保護ポリシーとしてVMMに渡しておく．本手法をFAT32に適用し，ファイル保護の実証及び性能実験を行った．

# Guest OS file protecting by virtual machine monitor

YOSUKE CHUBACHI,[†1] TAKAHIRO SHINAGAWA[†2]
and KAZUHIKO KATO[†2]

Modern desktop enviroments are often exploited by kernel-level rootkits. Then traditional access control in operating system(OS) can be disabled or subverted that causes extensive damage to OS. In this paper, we propose guest OS file protecting by virtual machine monitor(VMM), which can defend the files even if the OS is untrustworthy.

They have semantic gap between files on OS and on storage devices. Our proposal the access control policy made secure environment and the mapping of the gap. VMM enable file protection by using the map. We implemented the proposed protection for FAT32. We confirmed the protection and evaluated performance.

## 1. はじめに

情報化社会が進む今日，コンピュータやネットワーク，そしてその上に存在する情報資源に対しての脅威は年々増え続けている．これにより攻撃者は，攻撃プログラムの振る舞いがユーザに気付かれないように，オペレーティングシステム（OS）そのものを攻撃対象に加えた．トロイの木馬やカーネルルートキットと呼ばれるこの不正なプログラムは，OSを改変して自身の情報をユーザや他のプロセスから隠蔽するなど，OSカーネル内で不正な動作を行う．このようにOSが信頼できない状態に陥った場合，システムのセキュリティを保障することは難しい．

これらの攻撃からファイルを保護する方法として，OSが提供するアクセス制御が挙げられる．これにより不正なユーザやプログラムが，重要なシステムファイルを書き換えるといった不正な操作を行う事を阻止するものである．しかし，攻撃によってOSが改変を加えられた場合はそれらの機能が無効にされている可能性がある．このように攻撃者が完全にOSを掌握し，その機能すべてを利用されうる状態となったコンピュータに対して，従来のアクセス制御の正当性を保証することは困難である．

本研究では，ストレージデバイス上でのファイル情報に注目し，仮想マシンモニタ（VMM）のデバイスドライバにアクセス制御機能を組み込むことによりOSから独立したファイルアクセス制御を提案する．このアクセス制御は2つのフェーズに経て有効化される．始めにセキュアな環境でポリシを作成するプログラムを実行する”ポリシー作成フェーズ”がある．その後，作成された論理ブロックアドレス（LBA）ベースのポリシを用い，VMMがデバイスドライバのI/Oに対してコマンド発行の可否を制御する”アクセス制御実施フェーズ”でアクセス制御が有効になる．これにより，OSが信頼できない状態においても，VMMがreadonlyと既定したファイルの改ざんを防ぐことが可能になる．これに基づき，対象となるファイルシステムをFAT32としてVMMにファイル保護機能を実装し，その保護が実行中のWindowsにどのような影響を与えるか調査を行う．また，これによって明らかになった，VMMによるアクセス制御がOSに及ぼす影響について考察する．

†1 筑波大学 第三学群 情報学類
College of Information Science, Third Cluster of Colleges, University of Tsukuba
†2 筑波大学大学院 システム情報工学研究科 コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba

Vol.2009-ARC-183 No.31
Vol.2009-OS-111 No.31
2009/4/24

本稿では，ファイルとストレージとの間の“semantic gap”に対処する手法と，それに基づいて実装を行った VMM によるアクセス制御の評価と問題点を述べる．第 2 章では VMM でファイル単位のアクセス制御を行う際の問題点について述べ，第 3 章では VMM を用いたゲスト OS のファイル保護の方法について述べる．次に第 4 章では本システムの実装について，そしてアクセス制御の実証と性能の評価実験について第 5 章で説明する．

## 2. ストレージレベルのファイル保護

本章では，まず VMM でファイル単位のアクセス制御を実施する際のアクセスの粒度について述べる．次に，OS 上のファイルとストレージに格納されたファイルの“semantic gap”に説明し，それらを関連付けて VMM でファイル単位のアクセス制御をを行うアプローチについて述べる．

### 2.1 VMM を用いたアクセス制御の粒度

仮想機械（VM）で実行される OS の情報は，VM を提供する VMM がすべてを把握する事が可能である．例えば，アクセス制御におけるユーザ（プロセス）を，OS カーネル内のプロセス構造体を参照し識別することが可能である．しかし，悪意のあるプログラムに改ざんされた OS の情報や機能は改ざんされている可能性が高く，またその振る舞いも信頼できない振る舞いと考えることができる．これは，アクセス制御の主体（subject）となるユーザの識別が困難であり，OS のアクセス制御機能のような保護を行うためには情報が不足するということを意味する．

本研究では，デスクトップ環境における OS のファイル保護を目的とした．具体的には，OS のシステムファイルを不正に書き換える攻撃に対してファイル保護を行う場合を想定する．このとき，システムファイルに書き込みを行おうとしているプログラムが，ユーザ権限あるいはカーネルモードであっても等しく保護を実施しなければならない．これを実現するためにアクセス制御の主体を OS とし，OS の振る舞いすべてをアクセス制御の対象とする．このアクセス制御は VMM のストレージデバイスドライバで実現する．

### 2.2 資源の抽象度に関する問題

VMM は OS の抽象化された資源をそのまま扱う事ができず，また OS の機能に介入する事が困難である．ファイルやディレクトリは，ストレージ上ではデータとファイルの属性情報の二つに分けて保存される．それらはストレージ上の連続した領域に保存されるとは限らず，結果的にファイルアクセスが複数回の I/O に対応し，これが TOCTOU 競合条件となる可能性がある．

このファイルとストレージのギャップに関して，本研究ではまず，”ポリシ作成フェーズ”と”アクセス制御実施フェーズ”の 2 つのフェーズを設ける．そして，ファイルシステムとストレージ上のデータを関連付けるポリシメーカーを導入し，これをポリシー作成フェーズで実行する．ポリシメーカーはユーザ権限で動作するアプリケーションである．これはストレージを Raw-Device としてオープンして，これを対象となるファイルシステムの仕様に基づいて探索し，ファイルシステム上のファイル名をストレージの論理ブロックアドレス（LBA）とファイルの属性情報に関連付けを行う．

また，ファイルアクセスがデバイスドライバでは複数回の I/O となる問題については，ストレージへのデバイスドライバの read/write コマンド発行すべてを監視を行い対応する．これにより，ポリシで対象となっている LBA へのアクセスは必ず禁止とし，保護対象を確実に保護する．

## 3. システム設計

本研究で想定する脅威モデルと，アクセス制御モデルを述べる．またそれらに応じたアクセス制御ポリシの作成方法について説明し，また，実際にアクセス制御を行うために必要なポリシ作成フェーズと，そこで作成したポリシを用いて実際に保護を行うアクセス制御実施フェーズという 2 つのフェーズを用いた保護について述べる．

### 3.1 脅威モデル

本研究では，デスクトップ環境で利用されている OS の保護を目的としており，OS 1 つに対して，ユーザは 1 人となっており，またネットワークに接続して利用されているとする．OS には脆弱性が存在し，また管理者でもあるユーザがセキュリティ上の問題がある設定を行う可能性があるとする．その脆弱性や設定のミスにを突き，ネットワークを介して攻撃者が侵入した際にもファイルの保護を行えるようにしたい．このとき，アクセス制御を行う VMM には侵入されることはないとする．これは，VMM 自体の規模が小さいため脆弱性の数や性質などから，攻撃を受けにくいと考えられるからである．加えて，その OS が VMM で動作しているかどうかをユーザが知ることができないと仮定している． 以上から，**表 1** のような脅威モデルを想定する．

### 3.2 アクセス制御モデル

アクセス制御を行う際には，そのモデルとしてアクセスの主体（subject）とアクセス対象（object），そして，許可する行動（action）が必要である．本研究ではアクセス対象の単位を OS が管理するファイルとする．またアクセスの主体に関しては，2 章で述べた通り

Vol.2009-ARC-183 No.31
Vol.2009-OS-111 No.31
2009/4/24

**表 1** 想定する脅威モデル

| 想定する脅威 | 目的 | ターゲット |
|---|---|---|
| ウィルス・ワーム | OS の破壊や改ざん | システムファイル |
| トロイの木馬・ボット | OS の改ざん | システムファイル |
| ユーザの過失 | ユーザの過失によるシステムの破壊 | システムファイル |

ユーザと OS を一つのアクセスの主体とする．許可する行動は，デバイスドライバで認識できる行動が Read/Write のみであるため，これ以上の粒度でアクセス制御を行うことはできない．これを元に，以下のようなアクセス制御ポリシを利用できるようにした．

**基本的に読み書き可能（Default Allow）なポリシ**

(1) オペレーティングシステムがファイル A を読み込むことを禁止
(2) オペレーティングシステムがファイル B を書き込むことを禁止
(3) オペレーティングシステムがファイル C への読み書きすることを禁止

想定する脅威から，保護の対象は主に OS のシステムファイルの保護とする．

### 3.3 ポリシメーカー

ストレージ上のファイルは，ファイルシステムの仕様に沿って保存されている．ファイルシステムはファイルシステム固有の方式でファイルの情報を格納しており，これらは一般的に OS の管理情報であるファイルの属性情報，ストレージのブロック管理情報そして実データなどに分割されている．OS はファイルシステムへアクセスする際に OS が提供する API を用いてストレージ上の分割された情報を参照し，ファイルの情報を読み込んだりディレクトリの参照を行う．この API からは OS が必要としない情報，つまりストレージにおけるファイルの配置情報などは直接知ることができない．従って，ストレージ上のファイルの配置を知るためには，ストレージを OS がファイルシステムを扱う方法に沿って，それらを独自に探索する必要がある．

ファイルシステム上のファイル名を LBA とファイルの属性情報に関連付ける機構として，ポリシメーカーを導入する．これはストレージを Raw デバイスとして開き，ファイルシステムの仕様に基づいて，ファイル名と LBA, 属性情報を列挙するものである．

### 3.4 保護を有効にするための 2 フェーズ

ポリシメーカーはユーザーランドで動作し，ファイルとストレージを関連付けるために保護の対象となるストレージを Raw デバイスとしてオープンする．このように，ストレージ全体に対してすべてのアクセスできる必要があるため，ポリシメーカーは OS が信頼できる状態で VMM による保護を無効にして実行する必要がある．このポリシーを作成する環境をポリシ作成フェーズとする．加えてファイル保護を実施するとき，この一時的に OS が信頼する状況においてシステムのアップデートや設定などシステムファイルを更新する状況が起こりうる．このとき，OS を一時的に信頼してシステムファイルの更新を許可する必要がある．この状況では OS を信頼できる環境下に配置し，そこで作業を行うことでセキュリティを保つ．

VMM 上でゲスト OS を起動し実際にアクセス制御を行うフェーズを，アクセス制御実施フェーズとする．このフェーズはポリシ作成フェーズで作成したポリシを VMM に渡し，VMM が起動するところから開始される．このフェーズでは作成したポリシに基づき，VMM がデバイスドライバレベルでアクセス制御を行う．このとき，ゲスト OS に対して変更などは一切必要なく VMM 上で起動するだけで保護が有効になる．

## 4. 実　　装

ここまでに述べた提案に基づき，ファイル保護の実装をポリシメーカーと VMM 拡張の 2 つについて行った．ポリシメーカーは FAT32[1] を対象とし，C++と WindowsAPI を用いて作成した win32 アプリケーションである．FAT32 を対象とした理由は，デスクトップ環境で一般的な Windows を対象としたこと，FAT32 は元々セキュリティ属性が有効になっておらずセキュリティ強化の必要性があること，そして FAT32 はその仕様が公開されており，これに基づいた探索が可能であることからである．VMM はベースに BitVisor[2] を用い，その ATA[3] デバイスドライバを拡張することでファイル保護を実現した．ここでは，ポリシメーカと VMM 拡張のベースとなった VMM である BitVisor について述べ，その拡張について説明する．

### 4.1 ポリシメーカー

ポリシメーカーは FAT32 でフォーマットされたストレージを探索し，与えられたファイル名を元にそのファイルの配置情報と属性情報を列挙するプログラムである．まず，FAT32 について概略を述べ，その実装について説明する．

#### 4.1.1 FAT32

FAT（File Allocation Table）ファイルシステムとは，Microsoft 社が開発したファイルフォーマットである．特徴として，ストレージをクラスタという単位に分割し，File Allocation Table というデータテーブルを用いて管理していることが挙げられる．これによって，データをクラスタの連鎖を用いて表現し，ストレージの効率的な利用を計っている．

Vol.2009-ARC-183 No.31
Vol.2009-OS-111 No.31
2009/4/24

#### 4.1.2 ディレクトリエントリとデータ領域

FAT ではファイルを，ディレクトリエントリと呼ばれる 32byte のデータセットで管理している．これを**図 1** に示す．

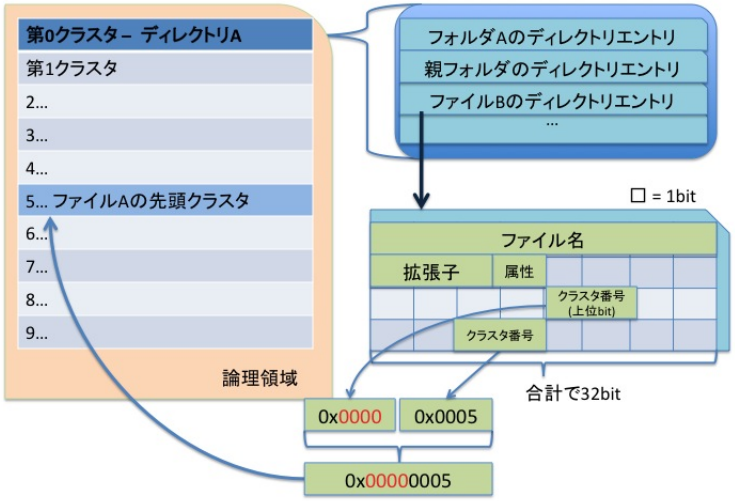


**図 1** ディレクトリエントリ

このディレクトリエントリに示されるクラスタが，データ領域に存在するファイルを示している．このデータ領域にファイルはクラスタ単位に分割されて格納される．Windows におけるフォルダは，同じようにディレクトリエントリとして格納され，そのフォルダの子が格納されているクラスタを示す．もしクラスタが 1 つで足りない場合，File Allocation Table を参照して次のクラスタを参照する．

#### 4.1.3 ポリシメーカーの実装

本実装で保護の対象とするのは，以下の二つである．

- ファイルの属性情報　（ファイル名，拡張子，アクセス日時など）
- ファイルのユーザーデータ領域　（データの中身）

この二つは FAT32 におけるディレクトリエントリの情報とユーザーデータの開始クラスタ番号，そしてユーザーデータのクラスタチェインで表現されている．ポリシメーカーはまず，ファイル名を用いて保護対象であるファイルのディレクトリエントリ（32bit）を探索してポリシとして保存する．そのとき，ディレクトリエントリが保存されているクラスタ番号も同時に保存する．次に，ディレクトリエントリに存在するユーザーデータ領域の先頭クラスタを保存し，クラスタチェインを追跡しながら連鎖しているクラスタ番号を保存する．これにより保護の対象は以下のように表現される．

- ファイルのディレクトリエントリと，そのエントリが保存されているクラスタ及びクラスタ内の配置情報
- クラスタ番号の集合

このポリシのクラスタ番号をセクタの集合に変換したものが，VMM のアクセス制御に用いるポリシとなる．

### 4.2 VMM 拡張

本実装は，BitVisor をベースとし，ATA デバイスドライバにアクセス制御を追加した VMM を用いた．まず，BitVisor と ATA デバイスドライバの read/write について説明し，アクセス制御機能の実装について述べる．

#### 4.2.1 BitVisor

BitVisor とは筑波大学が中心となって開発中の VMM である．その開発目的はデスクトップ OS のセキュリティ強化であり，BitVisor は OS とハードウェアの間で動作し，確実にセキュリティ機能が有効になるという組み込まれたセキュリティシステムを実現する．セキュリティ強化が目的であるため，複数の仮想マシンを動作させるなど一般的な VMM とは提供する機能が違うのも特徴であり，このため BitVisor ではハードウェアと OS は 1 対 1 でしか動作しない．また，現在サポートされている OS は，Microsoft Windows XP，Vista，Linux，そして FreeBSD である．

BitVisor は，Type1 ハイパーバイザ型の VMM であり，準パススルー（parapass-through）アーキテクチャの VMM である．準パススルーアーキテクチャとは，仮想化するデバイスを最小限に抑え，パフォーマンスと VMM としての大きさを抑えるアーキテクチャである．本研究では BitVisor の ATA デバイスの仮想化を用いて，デバイスドライバレベルのアクセス制御を実装している．これは，BitVisor が仮想的な ATA デバイスドライバをゲスト OS に提供し，ゲスト OS はそれを用いてストレージにアクセスを行う．このとき，BitVisor の実 ATA デバイスドライバにてゲスト OS が発行した ATA コマンドを監視し，場合によってはコマンドの発行を中止することができる．

Vol.2009-ARC-183 No.31
Vol.2009-OS-111 No.31
2009/4/24

#### 4.2.2 VMM におけるアクセス制御の実装

本実装では指定したファイルの領域に対して書き込みが発生した場合にそれをブロックするような実装を行った．これは，BitVisor の ATA ドライバ中にあり，ATA コマンドを発行する箇所にセキュリティチェックを行う関数を挿入することによってアクセスの可否を返す．例として OS がファイルに対して書き込みを試みたときのシステムの動作を**図 2** に示す． 図 2 中の番号に対応して以下のような流れでアクセス制御を行う．

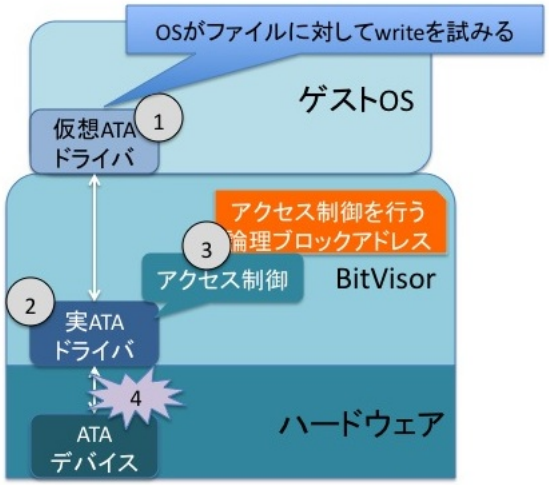


**図 2** アクセス制御の流れ

(1) BitVisor の ATA デバイスドライバに，OS の仮想デバイスドライバから write コマンドが送られる
(2) デバイスドライバが write を発行する直前にセキュリティチェック関数が実行される
(3) セキュリティチェック関数はポリシを参照し書き込みの可否を決定する
(4) チェックの結果に応じてデバイスドライバは write コマンドを発行／中止する

セキュリティチェック関数では書き込み先の LBA が禁止されていないかどうかをチェックし，禁止されていればコマンドの発行を許可しない．また，書き込み先の LBA がディレクトリエントリである場合，ディレクトリエントリが変更されていなければ書き込みを許可する．これは，保護の対象となるファイルのディレクトリエントリではなく，同じディレクトリの違うファイルが書き換えられる可能性があるためである．

## 5. 実　　験

本章では実装したポリシメーカーと拡張した BitVisor を用いて，実際にアクセス制御が行えるかどうか実験を行った．実験環境と方法，結果について述べその実験結果について考察を行う．また，この実装の性能評価を行った．

### 5.1 アクセス制御の実証実験

拡張を行った BitVisor を用いて，アクセス制御機能の実証実験を行った．

#### 5.1.1 実 験 環 境

実験を行った環境を以下に示す．

- OS:Microsoft Windows XP SP3
- CPU:Intel Core2Duo E6850 3.0GHz
- メモリ: 2GB
- HDD: MTRON MSP-SATA7035 （SSD）IDE モード

#### 5.1.2 実 験 方 法

ファイルと論理ブロックアドレスの対応付けをポリシメーカーによって行い，リスト 1 のようなポリシを作成した．それを用いて BitVisor を実行し，その上で Windows XP を起動してエディタでファイルにアクセスする．このときの Windows XP の挙動を調べた．

リスト 1　target_lba_info.h

```
#define TARGET_DEVICE_ID 0x0000
#define TARGET_HOST_ID 0x02

#define DE_LBA eefe00
#define DE_NUM 4
u64 UserDataPolicies[5] = {
  53375,
  31,
  31263,
  31,
  0};
u32 UserDataPoliciesSize = 4;
u8 deValue[33] = {0x54,0x41,0x52,0x47,0x45,0x54,0x20,0x20,0x54,0x58,0x54,0x20,0x18,
0xc0,0x88,0xa5,0x2c,0x3a,0x6a,0x3a,0,0,0x2c,0xa4,0x6a,0x3a,0xca,0x2,0x77,0x58,0,0,};

```

Vol.2009-ARC-183 No.31
Vol.2009-OS-111 No.31
2009/4/24

```
16 #endif
```

まず，DE_LBA，DE_NUM，deValue が，それぞれ対象となるファイルのディレクトリエントリが保存されている LBA，セクタを頭から数え何番目のディレクトリエントリなのかを示す DE_NUM，また，その値である deValue の 3 つである．UserDataPolicies は，データが格納されるクラスタを示す．データは連続的に書き込まれることが多いので，ベースとなる LBA から n カウントというペアを列挙した配列を用いてクラスタを表している．TARGET_DEVICE_ID は IDE のマスター／スレーブを表す ID であり，TARGET_HOST_ID はストレージが接続されている ATA チャンネルを指定する．

**実験の手順**

(1) BitVisor を起動し，そのまま WIndows XP を起動
(2) 対象物理ドライブのファイルをエディタで開く
(3) エディタから文字を追記し，保存を試みる

#### 5.1.3 実 験 結 果

見かけ上は書き込まれたように見えるが，数十秒から 1 分以内に Windows が**図 3** のようなエラー通知をした．その後，OS を再起動するとその書き込みが無効になっており，ファイルの属性情報も保護を行う前の更新内容となっていた．

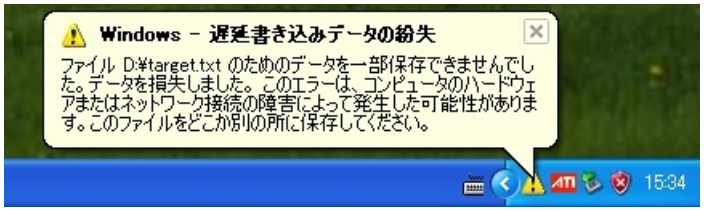


**図 3** Windows のエラー通知

#### 5.1.4 考　　察

OS が持つファイルキャッシュが情報を保持するため，書き込みを無効にした場合も見かけ上は書き込まれた事になっていると考えられる．また，書き込み遅延という表示が行われる事について BitVisor のログを参照すると，何度も繰り返して書き込もうとしていることが判明した．

### 5.2 性能評価実験

拡張を行った BitVisor のオーバーヘッドを，ベンチマークソフトを用いて性能評価を行った．

#### 5.2.1 実 験 環 境

実験を行った環境を以下に示す．

- OS:Microsoft Windows XP SP3
- CPU:Intel Core2Duo E6850 3.0GHz
- メモリ: 2GB
- HDD: MTRON MSP-SATA7035 （SSD）IDE モード

また，性能を比較するために，実験の対象を以下のものとする．

(1) 拡張された BitVisor + Windows XP（Extend BitVisor）
:本研究で実装した，拡張された BitVisor 上で，XP を起動
(2) BitVisor + Windows XP （Normal BitVisor）
:本研究の実装のベースとなった BitVisor 0.7 上で，XP を起動
(3) Windows XP (Native)
:直接 XP を起動

#### 5.2.2 ベンチマークについて

ベンチマークには Crystal Mark 2004R4 の機能である HDD ベンチマークを用いた．このベンチマークでは，次に挙げる処理の性能を測定できる．

- Sequential Read ( MB/s )
- Sequential Write ( MB/s )
- RandomRead512K ( MB/s )
- RandomWrite512K ( MB/s )
- RandomRead 64K ( MB/s )
- RandomWrite 64K ( MB/s )

テスト項目は連続領域の読み書きとランダムな読み書き，またそのサイズについてである．また，本実験では SSD を対象としている．SSD を従来のハードディスクドライブと比較したときの特徴として，連続領域の読み書きと，ランダム領域への書き込みはハードディスクより遅く，ランダム領域での読み込みは 2 倍ほど早くなっている．

Vol.2009-ARC-183 No.31
Vol.2009-OS-111 No.31
2009/4/24

#### 5.2.3 性能評価

性能評価の結果を**表 2** に示す．また，そのグラフを**図 4** に示す．

**表 2** ベンチマーク結果

| 評価項目 | Native | Normal BItVisor | Extend BitVisor |
|---|---|---|---|
| SequentialRead( MB/s ) | 79.85 | 70.28 | 69.98 |
| SequentialWrite( MB/s ) | 71.13 | 59.21 | 59.56 |
| RandomRead512K( MB/s ) | 79.62 | 70.14 | 70.05 |
| RandomWrite512K( MB/s ) | 38.24 | 37.61 | 37.72 |
| RandomRead64K( MB/s ) | 75.20 | 63.02 | 62.91 |
| RandomWrite64K( MB/s ) | 15.95 | 15.73 | 15.34 |

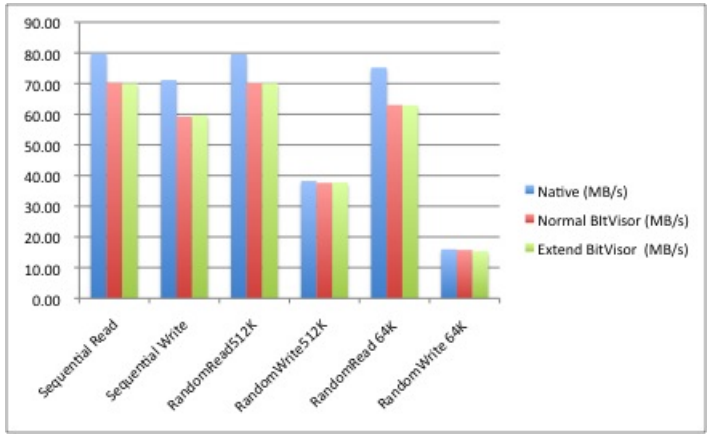


**図 4** 性能評価

#### 5.2.4 考　　察

拡張を行う前の BitVisor と比較して，性能の変化は見られない．実装ではデバイスドライバレベルでの処理として，特定のディスクに対しての書き込みすべてをチェックし論理ブロックアドレスのある範囲からある範囲までにアクセスしなければアクセスを許可するような処理を行う．結果からデバイスドライバでのアクセス制御が性能に与える影響は少ないといえる，

## 6. 関連研究

本研究の目的は OS から独立したアクセス制御を行うことにより，信頼できない OS，信頼できないユーザからファイルを保護することである．このような OS から独立したアクセス制御を行う研究として，SVFS[4] がある．SVFS は仮想マシンである Xen[5] を用いて，ファイルの管理を行う OS とユーザが用いるゲスト OS を分離し，ゲスト OS がファイルにアクセスする際にはファイル管理 OS がそのアクセスコントロールを行う．1 つの PC を 1 つの OS で利用する際も，ゲスト OS，ファイルシステム管理用の OS，システム全体の管理を行う OS の 2 つを起動する必要がある．この SVFS と TPM を用いて，正しい鍵を持ったユーザが閲覧のみ可能とするファイルシステム VOFS[6] が提案されている．

また，これらのファイルシステムを外部に設ける方法に加え，ユーザ認証の利便性の向上を図った研究として SAccessor[7] がある．SAccessor では NFS を用いてファイルサーバとなる OS が横断的に認証を提供し，OS の外部で認証とアクセス制御を行う方法を提案している．これらの研究では，ファイルシステムを管理する別の OS を用いており，この OS が信頼できない状態になる可能性がある．また SVFS はゲスト OS のファイルシステムを変更しなければならず，SAccessor についてもファイルキャッシュをクリアするために OS に依存した設計となっている．これに対して，本研究では基本的に OS には依存せず，ファイルシステムの仕様に沿って実装を行う事で，そのファイルシステムを利用できるすべての OS に対して変更を加える事無く保護を有効にすることができる．またこれらは，VMM やネットワークを介してストレージを利用するため，そのオーバーヘッドは大きいが，本システムではデバイスドライバでこれを行うため比較的ネイティブに近い性能を保つことができる．

## 7. まとめと今後の予定

本研究を進めるにあたっての問題意識と目的について述べ，VMM による LBA ベースのアクセス制御を行う際の問題点について述べ，その解決のためのアプローチを示した．また，VMM によるファイル保護を行うシステムの設計のために脅威モデルやアクセス制御モデルを設定し，そこからポリシメーカーの導入，そして 2 フェーズによるファイル保護など，システムの設計を行った．その設計に基づきポリシメーカーと VMM の拡張を実装した．実装したシステムの実験を行い，ファイルの保護が有効になるということを確認した．また，性能についてベースとなった BitVisor やネイティブで動作させた OS と比較し，性能の評価を行った．最後に関連する研究について述べ，本研究の優位性，新規性について議

Vol.2009-ARC-183 No.31
Vol.2009-OS-111 No.31
2009/4/24

論した．

## 参 考 文 献

1) MicrosoftCorporation: FAT32 File System Specification. http://www.microsoft.com/japan/whdc/system/platform/firmware/fatgen.mspx.
2) 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, *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, New York, NY, USA, ACM, pp.121–130 (2009).
3) T13, T.C.: AT Attachment. http://www.t13.org/.
4) Zhao, X., Borders, K. and Prakash, A.: Towards Protecting Sensitive Files in a Compromised System, *SISW '05: Proceedings of the Third IEEE International Security in Storage Workshop*, Washington, DC, USA, IEEE Computer Society, pp. 21–28 (2005).
5) 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, *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, New York, NY, USA, ACM, pp.164–177 (2003).
6) Borders, K., Zhao, X. and Prakash, A.: Securing sensitive content in a view-only file system, *DRM '06: Proceedings of the ACM workshop on Digital rights management*, New York, NY, USA, ACM, pp.27–36 (2006).
7) 滝澤裕二，光来健一，千葉　滋，柳澤佳里：SAccessor：デスクトップ PC のための安全なファイルアクセス制御，情報処理学会論文誌:コンピューティングシステム（ACS），Vol.1, No.2, pp.275–285 (2008 年 8 月).