2W-06

準パススルー型ハイパーバイザを利用したブロックストレージの書き込み監視システム

都築　卓馬†　　海野　友希†　　平野　学†

独立行政法人国立高等専門学校機構

豊田工業高等専門学校 情報工学科†

1. はじめに

現在，コンピュータに関する犯罪やインシデントが発生した際には，現場のコンピュータに残っている証拠を回収し，それらを解析することで，過去のタイムラインを推定する手法が用いられている．しかしながら，重要度の高いコンピュータシステムの場合には，そのような事後対策ではなく，事前にある種の監視システムを動作させておくことで，犯罪の抑止効果が狙えるほか，インシデント発生後のタイムラインを正確に再現できるようになるはずである．

本稿では，組織で特に重要な機密を扱うユーザに対して，監視システムの存在を対象ユーザが承認していることを前提に，そのユーザのコンピュータ上の振る舞いを一定期間記録する監視カメラのようなシステムを検討する．本稿では準パススルー型ハイパーバイザである BitVisor と呼ばれる仮想化ソフトウェアに，HDD や SSD のようなブロックストレージへの書き込みを監視する機能を組み込んで，定期的に解析用の分散ファイルシステムへ転送させるようにしたシステムの例を示す．

2. 準パススルー型ハイパーバイザ

本稿ではユーザが直接操作するコンピュータ端末の挙動を監視する機構を検討する．この監視はアプリケーション，OS，ハイパーバイザ（仮想化ソフトウェア），ハードウェアの各層で実現可能である．しかし，rootkit などが多く流通している OS 層では監視機能を迂回される可能性が高まるため，本稿では OS と比較してソースコードが小規模で脆弱性を見つけにくいと考えられるハイパーバイザを監視機能の実装に利用する．仮想化を実現するハイパーバイザにはサーバ向けに利用されている VMware ESX や KVM などがあるが，それらのハイパーバイザは監視対象以外のハードウェアも仮想化するため，特定の入出力を監視するには余分な負荷がかかってしまう．この問題を解決するため，品川らは BitVisor と呼ばれるセキュリティに特化したハイパーバイザを開発した[1]．品川らの BitVisor は，監視を必要とする入出力だけを仮想化し，それ以外の入出力をそのまま通過させる準パススルー・アーキテクチャを採用することで，同時に実行できるゲスト OS は一つだけに制限されるものの，仮想化による性能の低下を最小限に抑えている．本稿の目的はブロックストレージの入出力を監視することが目的であるため，性能と安全性を考慮して，準パススルー・アーキテクチャの BitVisor を採用した．

3. ブロックストレージへの書き込み監視

ブロックストレージへの書き込み監視の目的は，監視カメラの動画のように，書き込まれたブロックデータ（4 KiB ブロック）を，タイムスタンプと Logical Block Address (LBA)とともに，一定期間記録しておき，”再生”が必要な時に，指定された時刻の HDD や SSD のディスクを Read only の状態で復元することである．著者らは仮想マシンモニタの Xen に，上記の記録と再生の機能を実装し，監視カメラのように，仮想マシンのディスク書き込みを 4 KiB 単位でタイムスタンプとともに逐次記録し，必要な時に指定された時刻のディスクの状態を再生するシステムを開発した [2][3]．このシステムは Xen での実装であるためサーバ以外のクライアント用途には性能面や安全性の面で適していない問題があった．そこで，本稿では記録の機能を BitVisor に移植するものとした．BitVisor ではブロックストレージへの書き込みを（4 KiB のブロックデータ，LBA，タイムスタンプ）の組で記録して，一定期間ごとに解析用の分散ファイルシステムへ転送して蓄積する．著者らは[3]の研究で，Hadoop を用いて大量のブロックデータから高速に特定ファイルを検出するシステムを提案した．本稿で提案する BitVisor から得られた監視データは，このシステム[3]で解析できるようにシステムを設計した．

A Block Storage Surveillance System Based on a Parapass-through Hypervisor.

† Takuma Tsuzuki, Tomoki Umino, and Manabu Hirano, Department of Information and Computer Engineering, National Institute of Technology, Toyota College.

4. 設計

本稿で開発するシステムの構成を図 1 に示す．本稿の監視システムはシリアル ATA 接続された HDD や SSD を対象とし，それらの装置へのデータの書き込みだけを記録用バッファに記録し，それ以外の入出力はそのまま通過させる．同時に，書き込みされたデータは定期的に記録用バッファからネットワーク経由で同一 LAN にある Hadoop Distributed File System (HDFS) へ転送する．HDFS を用いるのは Hadoop や Spark などの多くの解析プラットフォームから利用できるからである．HDFS 自体にはネットワーク経由でファイルを操作する機能がないため，WebHDFS と呼ばれる HTTP サーバを利用する．WebHDFS は現行執筆時点で最新版の Hadoop 2.7.1 に標準で含まれる，REST API を用いて遠隔から HDFS のファイルを操作する機能である．

5. 実装

BitVisor で HDD や SSD のようなブロックストレージの監視をするため，本稿では AHCI (Advanced Host Controller Interface) と呼ばれるシリアル ATA 規格の記憶装置を扱うコントローラを監視ポイントとした．BitVisor の提供する AHCI の準パススルードライバを改良することで，書き込みデータを監視する．本稿では BitBucket にある BitVisor のソースコード(2015 年 12 月 2 日時点の最新版)を用いて開発を行っている．具体的には，BitVisor のソースコードの drivers/ata/ahci.c にある DMA のメモリ領域を BitVisor のシャドウバッファへコピーする処理に変更を加え，書き込まれたデータを 4 KiB 単位で LBA 番号と時刻情報とともに記録する機能を実装する．記録された 4 KiB ブロックのデータを WebHDFS サーバに転送する処理は，BitVisor の提供するスレッド機能を用いて定期的に呼び出されるように実装する．WebHDFS プロトコルでのデータ転送の処理は，BitVisor に組み込まれている lwIP と呼ばれる軽量の TCP/IP スタックを用いて実装する．

6. 考察とまとめ

面らはベアメタルクラウドでの OS 配備を高速におこなう BMcast 機構を提案した[4]．BMcast は BitVisor の IDE と AHCI のドライバを，ATA over Ethernet (AoE) によるデータ転送に対応させた．本稿のシステムは BMcast のような高度な AoE は実装していないが，lwIP で実装した REST API で分散ファイルシステムへ 4 KiB

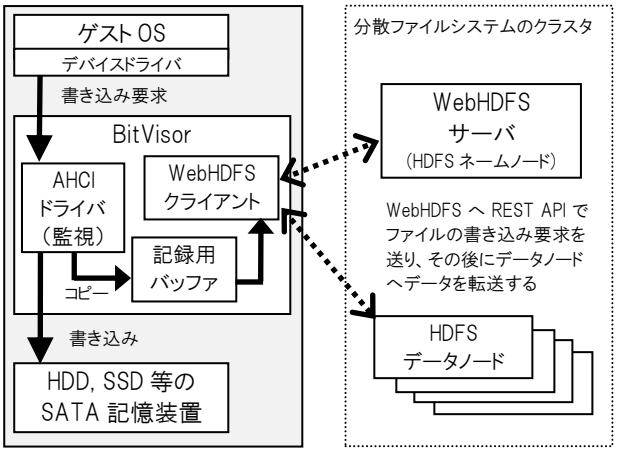

図 1 BitVisor を用いた監視システムの全体構成

ブロックを LBA と時刻情報とともに転送する設計である．記録バッファの容量と HDFS へのデータ転送速度が入出力のボトルネックになる可能性がある為，実環境での評価実験が必要である．

謝辞

BitVisor の開発チームよりたくさんの貴重な助言を頂きました。本研究は JSPS 科研費 26330168 の助成を受けたものです。

参考文献

[1] Shinagawa, T. et al.: BitVisor: a Thin Hypervisor for Enforcing I/O Device Security, In Proc. of the 2009 ACM SIGPLAN /SIGOPS International Conference on Virtual Execution Environments, pp.121-130, (2009).
[2] 小川拡，平野学：仮想計算機モニタを利用したコンピュータフォレンジックスのための補助記憶装置のデータの保全と回復のシステム，情報処理学会研究報告 CSEC, 2013.16 (2013).
[3] Hirano, M., Takase, H., and Yoshida, K.: Evaluation of a Sector-Hash Based Rapid File Detection Method for Monitoring Infrastructure-as-a-Service Cloud Platforms, In Proc. of the Availability, Reliability and Security (ARES), pp.584-591 (2015).
[4] Omote, Y., Shinagawa, T., and Kato, K.: Improving Agility and Elasticity in Bare-metal Clouds, In Proc. of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'15), ACM, pp.145-159 (2015).