Vol.2013-DPS-154 No.16
Vol.2013-CSEC-60 No.16
2013/3/14

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

小川拡[†1]　　平野学[†1]

近年，官公庁や企業がコンピュータに関する犯罪や法的紛争に巻き込まれた場合に，電子データを証拠として扱う必要性が増している．特にサーバ OS はクラウドコンピューティング環境で実行されることが増えており，仮想環境のための新たなデータ保全システムが求められていると考えられる．本稿では仮想計算機モニタで仮想化された補助記憶装置のデータを紛争解決に活用することを目指したシステムを提案する．このシステムでは書き込もうとした内容の履歴を時刻とともにすべて保管し，分析時に任意の時刻の状態でデータを再現する．提案システムは仮想計算機モニタを利用しているためゲスト OS が攻撃を受けても継続して動作し，ホスト OS に保存された保全データの改竄を防ぐことができる．提案システムは仮想計算機モニタのデバイスドライバを拡張する形で実装している．これにより複数の時刻の状態の仮想ディスクを同時にマウントすることができ，従来の解析ソフトウェアや Linux のシステムツールを利用して，違う時刻のディスク間の差分を分析できるようになった．本稿では上記の特徴を活かしてログファイルの改竄時刻を検出する例を示す．本稿では提案システムの設計と実装を示す．続いて提案システムの補助記憶装置の使用量と転送速度を測定した結果を示す．最後に今後の課題について考察する．

A Data Preservation and Restoration System of Storage Devices using Virtual Machine Monitor for Computer Forensics Investigation

HIROMU OGAWA[†1]　MANABU HIRANO[†1]

When companies and governmental organizations are involved in criminal or civil trials, they have to submit their digital data as evidence. In conventional cases, organizations submit physical hard disk drives as evidence. However, many recent business systems are constructed on cloud computing environments. In the cloud environment, virtual storage devices are not equivalent to physical storage devices. Therefore, we need a novel system of computer forensic investigation for virtualized environments. In this paper, we propose a data preservation and restoration system using virtual machine monitors. Our system preserves all of input and output data for virtual storage devices with the time. The proposed system can restore states of the target storage device of any specified time. We show some example usages of the proposed system like detection of the time of an incident that an attacker tampers with a log file in a virtual machine. The proposed system continue to run the preservation function even if the guest OS was compromised. In addition, the proposed system is implemented as tap software of virtual device drivers. Therefore, we can mount multiple virtual drives at the same time and the mounted virtual drives can have the state of each specified time. We can use conventional forensic software and Linux system tools for their analysis. This paper shows the result of performance evaluation of the prototype system. Finally, we discuss the future work and open issues.

1. はじめに

近年，官公庁や企業でのコンピュータの活用は当たり前のものになりつつある．それに伴い，重要な情報をコンピュータ上で取り扱う機会も増えており，コンピュータに関する犯罪や法的紛争において電子的な情報が証拠として使われる機会が増えている．コンピュータフォレンジックスは,情報の完全性を保護しながら,データを厳密に保管し，証拠データの識別，収集，検査，および分析を科学的手法によって行うことである[1].

本研究では，情報を完全な状態で確実に保全し，犯罪捜査時に必要な情報を取り出し，証拠を分析しやすい環境を実現するために,補助記憶装置への書き込み履歴を保存し，任意の時刻の状態を復元するシステムを設計，実装する．本提案では，仮想計算機モニタによって補助記憶装置へ書き込もうとした情報をすべて保全することで，OS 上で悪意のあるソフトウェアがファイルを削除・改竄した場合にも検出でき，攻撃を受ける前の情報を取り出すことも可能なため,コンピュータ上で扱った情報を確実に保全できる．犯罪者やマルウェアの中には証拠となるファイルやアクセスログを改竄して証拠隠滅を図るものもあるが，そのような場合にも補助記憶装置への書き込み内容を仮想計算機モニタのレベルで時系列に保全することで隠滅行為がいつ起きたかを特定できる可能性が高くなる．

提案システムで利用する仮想計算機モニタは，ハードウェアを疑似的に再現する環境を作り出すソフトウェアである．仮想計算機モニタは，ゲスト OS とは分離されることを目指して実装されているため，原則として OS 上での動作に影響を受けない．そのため，もし仮想計算機モニタの制御をゲスト OS の利用者から隔離することができれば，悪意のある利用者がシステムを停止させて履歴の保存を回避することはできなくなるはずである．

提案システムが想定するのは，企業や官公庁のような組織が管理するサーバシステムである.特に近年はサーバ OS をクラウドコンピューティング環境で実行させることが増

†1 豊田工業高等専門学校
Toyota National College of Technology

えている．特にコンピュータの操作履歴やアクセス日時が記録されているログファイルを保全することは重要である．ログファイルは不正アクセスを受けた際に証拠として提出されるからである．法的証拠として利用する為にはアクセスログを改竄されていない状態で保全するシステムを組織的に業務に組み込んで運用する必要がある．提案システムは仮想計算機モニタを利用しているため，クラウドコンピューティング環境で実行されるサーバ用途に適用しやすいと考えられる．

コンピュータフォレンジックスの性質上，コンピュータで扱う全てのデータを保全の対象とすることが理想的だが，頻繁に書き換えられるシステム関係のデータを対象としてしまうと，保全に必要とされる補助記憶装置の容量が膨大になってしまう．一方，ログファイルは基本的に追記のみを行うため，容量の問題が起きにくい特徴がある．本研究の提案はシステム領域を含むすべてのディスクに対する履歴保全が可能な設計である．しかし本稿では性能面を考慮し，第一ステップとしてログファイルを対象とした保全と回復のシステムの実現を目指すものとする．

2. 関連研究

鶴田らの研究[2]では，追記のみを許可するファイルシステムとヒステリシス署名を用いたログファイルの改竄を防止し，かつ，改竄を検出できるシステムを提案している．この手法では追記以外の操作を制限したファイルシステムを実装し，ログファイルの出力先を追記のみを許可するファイルシステムに指定することで，ログファイルの改竄を防いでいる．加えて，追記するデータに耐タンパデバイスを用いたデジタル署名を施すことで，補助記憶装置を持ちだされた場合や，ファイルシステムが攻撃を受けた場合にも改竄を検出できる．デジタル署名にはヒステリシス署名を用いて部分的な改竄を防いでいる.ヒステリシス署名は，図 1 に示すように複数のデータの原本があるとき，1 つ前の原本に対する署名を次の原本の署名に含めるものである．データを改竄する場合には署名の連鎖構造をすべて改竄しなければならないため，長期間に渡り安定して改竄を検出できる．このシステムでは，ログの改竄を検出することは可能なものの，ハッシュ関数の一方向性のため，改竄によって失われた情報を復元することはできない．攻撃者によってログを破壊され，証拠としての利用を困難にされてしまう攻撃に対する対策が必要である．

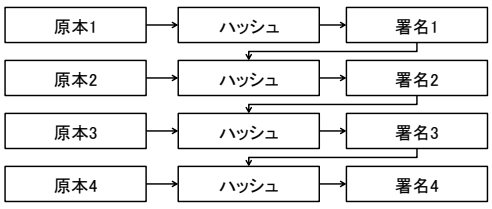

図 1 ヒステリシス署名

高田らの逃げログの研究[3]では，原本と同じ内容のバックアップファイルをファイルシステム上に複数作成し，原本を含めたすべてのファイルを定期的に監視して改竄を検出する．改竄が検出された場合には，他の正常なバックアップを用いてファイルを復元する．改竄検出後にバックアップから改竄前の状態を復元するため，攻撃を受けても正常なログファイルが残り，分析に利用できる．原本とすべてのバックアップを改竄された場合にはログの改竄を許してしまうが，バックアップは任意の数を作成でき，ファイルシステム上で定期的に別のディレクトリへ移動し，ランダムに名前が変化するため，短時間に原本とすべてのバックアップを改竄することは困難である．しかし，同一のログファイルを複数作成するため，処理の負荷が増大するという課題がある．

鶴田らの提案するシステムは，ファイルシステム，高田らの提案するシステムは，アプリケーションに組み込むライブラリとして実現されている．OS がルートキット等によって攻撃を受けた場合にはログ保全システムが影響を受ける可能性があり，正常に動作することを保証できない．

滝澤らの SAccessor の研究[4]では，作業用と認証用の仮想計算機を用意することで，攻撃を受けやすい作業用の仮想計算機からファイルアクセス制御機能を分離し，作業用の OS が攻撃を受けてもアクセス制御に影響しないシステムを実現している．しかし，認証用 OS だけでなく作業用 OS に対しても変更を必要とする．

忠鉢らの研究[5]では BitVisor と呼ばれる仮想計算機モニタを利用してファイル入出力を監視し，予め指定した領域へのアクセス制限を実現している．BitVisor は Xen よりもさらに小規模な仮想計算機モニタである．ホスト OS を必要としないハイパーバイザとして実装されているため，高速での動作が期待できるが実装の難易度が高い．BitVisor はクライアントのセキュリティ強化に適しているが同時に一つの OS しか動作させることができないためサーバ用途にはそのままでは利用できない．

Joel らの研究[6]では押収した補助記憶装置から高速に目的のファイルを検出するシステムを実現している．既知のファイルの検索はコンピュータ犯罪捜査で一般的な処理である．Joel らの研究では補助記憶装置のセクタ単位でハッシュ値を計算し，あらかじめ作成してある探索対象のファイルのハッシュデータベースと照合することでファイルシステムに依存しない高速な検索を実現している．本稿の提案もセクタを単位として処理を行うことで OS やファイルシステムに依存しない処理を実現する．

3. 提案システムの目的

本提案では仮想計算機モニタ上で動作する OS が補助記憶装置へ書き込もうとした時に，データの書き込み時刻とデータを全て保全する.保全したデータを分析する際には，

指定された時刻をパラメータとして与えて補助記憶装置の状態を復元する．同時に複数の時刻の状態を比較し，補助記憶装置に対して行われた操作履歴を調査できるシステムを目指す．加えて，仮想計算機モニタを用いることで，攻撃者に保全データを改竄・削除される可能性を低くする．

3.1 書き込みの保全

通常の補助記憶装置では，既に情報が書き込まれた領域に対して書き込みを行うと，図 2 の左に示すように，古い情報は上書きされて消えてしまい，証拠として利用できなくなってしまう．通常の犯罪捜査ではハードディスクを押収し（データの保全），データの複製を作成して解析を行っている．ファイルシステム上で消去されたデータを復元するために専用ソフトウェアを用いてファイルシステムのメタデータやセクタに残っている上書きされていないデータをもとに，可能な限りの類推により削除されたデータを復元する．同一セクタに新しいデータが上書きされない限り，ある程度のデータ復元は可能であるが，完全に該当セクタが上書きされてしまうと，以前にあったデータの復元は通常の解析方法では不可能である．

一方，Amazon EC2 のような IaaS（Infrastructure as a Service）でサーバ OS を実行している場合，ハードディスクは物理的には存在するものの，各サーバ OS に対応するハードディスクはイメージファイルとして存在することになる．このため物的な証拠としてハードディスクを押収していた従来の犯罪捜査とは異なる観点でのデータ保全を行う必要が生じる．例えば分散ファイルシステムを用いている場合は一つの物理ハードディスクに複数のテナントのデータが断片的に含まれている可能性があり，物理的にハードディスクを押収すること自体が難しくなることが考えられる．

本提案では保全対象を仮想化されたハードディスクのイメージファイルとする．更に，仮想化の特性を活かして，図 2 の右に示すように，書き込みを行う場合にも古い情報を削除せず，古い情報を残して情報を保全する機能を保全システムに組み込むものとした．保全する情報には時刻情報を付与し，復元時に時刻を指定できるようにする．これにより犯罪者が証拠を改竄や削除した場合に，証拠データを復元できるようにする．

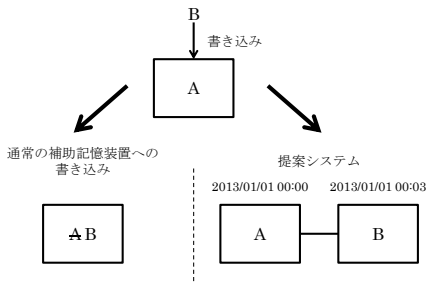

図 2 書き込みの保全（左は従来方法，右は提案システム）

3.2 状態の復元

保全データに付与された時刻情報を参照し，指定された時刻の状態を示す情報を選択して読み出す．分析時には，複数の時刻の状態を復元して比較を繰り返すことで，情報が改竄された時刻を特定できる．また，改竄される前の状態を復元することもできる．

4. 実現方式の検討

本節では，補助記憶装置の保全を実現する方法として，アプリケーションレベルで実現する方法，カーネルレベルで実現する方法，仮想計算機モニタで実現する方法，の 3 つについて，主に攻撃の受けやすさとファイルシステムの情報にアクセス可能かどうかの 2 点について比較検討する．

アプリケーションレベルで保全システムを実現する場合，ファイルシステムをユーザ空間のプログラムで実現する仕組みを利用する．Linux の場合標準で Filesystem in Userspace（FUSE）と呼ばれるカーネルモジュールが組み込まれている．FUSE を用いて実現されたファイルシステムにアクセスしようとした場合，FUSE のカーネルモジュールは入出力要求をユーザ空間で動作するプログラムに転送しユーザ空間で処理を行う．この方法は開発が容易であるが，ルートキット等によって OS が攻撃を受けて管理者権限を取得されてしまうと，ユーザ空間で動作するプログラムは停止されてしまう可能性がある．

カーネルレベルで監視システムを実現する場合，ファイルシステムの読み書きに関するシステムコールを改造する方法とデバイスドライバを改造する方法が考えられる．入出力されるデータは最終的にデバイスドライバが外部記憶装置を制御することによって実現されているため，デバイスドライバを改造することでデータの入出力処理を捕捉できる．システムコールを改造する場合，ファイルシステムの情報を参照できる．対して，デバイスドライバで実現する場合はファイルシステムの情報を利用することはできない．上記のどちらの方法でもルートキット等を用いて OS やデバイスドライバの脆弱性を攻撃された場合にはシステムを無効化されてしまう可能性は排除できない．

仮想計算機モニタは，OS とハードウェアの間で動作するため，OS からハードウェアへのデータや制御の入出力をすべて捕捉することができる．仮想計算機モニタにシステムを実装する場合，特定の OS のファイルシステムに依存しないため，複数の OS に対して同一システムを適用できる利点がある．仮想計算機モニタに脆弱性が存在した場合には，監視システムを無効化されてしまう可能性があるのは他の仕組みで実現した場合と同様である．しかし一般的に仮想計算機モニタは OS と比べると攻撃の難易度が高いと考えられる．本稿の提案システムはクラウドコンピューティングの基盤技術として仮想計算機モニタが利用されていることも考慮して仮想計算機モニタを採用する．

5. 仮想計算機モニタを介した入出力処理

本提案では，Xen の準仮想化を利用し，blktap[7]で補助記憶装置への入出力を捕捉する．blktap は，Xen でのブロックデバイスへの入出力の処理を拡張するための仕組みである．準仮想化は完全仮想化よりも性能面で有利であることが多いためIaaSにおいても利用されている．Xen と blktap を利用した入出力の流れを図 3 に示す．Xen では，管理用の特殊な仮想計算機をドメイン 0，それ以外の通常の仮想計算機をドメイン U と呼ぶ．ドメイン U は，基本的にハードウェアに直接アクセスできないため，ドメイン U では，デバイスドライバの代わりにスプリットデバイスドライバのフロントエンドが動作しており，補助記憶装置への入出力要求は，Xen の共有メモリを通じてドメイン 0 にあるスプリットデバイスドライバのバックエンドに送られて処理される．blktap を利用する場合は，スプリットデバイスドライバのバックエンドに渡った要求がさらに blktap のデーモンプログラム（tap-ctl）に渡され，tap-ctl に組み込まれた tapdisk ドライバが実際の入出力処理を行う．tapdisk ドライバは，用途に応じて複数実装されている．仮想計算機モニタの起動時やマウント時に，使用する tapdisk ドライバを指定することができる．

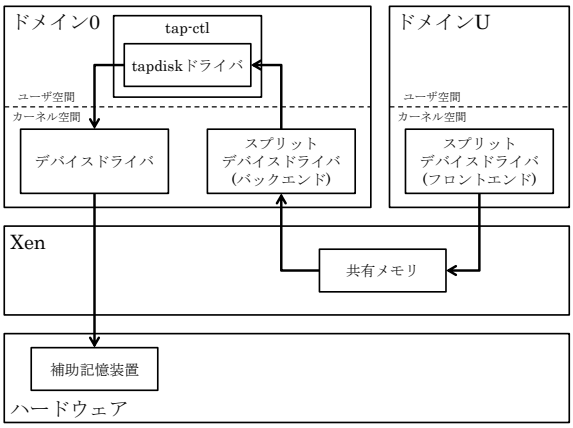

図 3 blktap を利用した入出力の流れ

6. システム設計

提案システムは Xen の blktap を利用した tapdisk ドライバとして実装し，保全する情報はドメイン 0 に保存する．本節では保全する情報のデータ構造と，保全機能と復元機能の具体的な実現方法を述べる．

6.1 データ構造

提案システムのデータ構造を図 4 に示す．データは片方向リンクリストになっており，リストの各エントリは 1 回の補助記憶装置への書き込みを表現している．リストの先頭にあるエントリを補助記憶装置への最新の書き込みとし，時系列に書き込みが並ぶ．リストにエントリが 1 つも存在しない場合は一度も書き込みが行われていないことを示す．

リストは補助記憶装置のセクタまたはファイルシステムのブロックごとに管理される．セクタのサイズは補助記憶装置の種類によって異なり，ハードディスクや SSD の場合は 512 バイト，CD や DVD の場合は 2048 バイトである．ブロックのサイズはファイルシステムや設定によって異なるが，Windows で利用される New Technology File System（NTFS）や Linux で一般的に利用される ext4 では 4096 バイトである．ブロックを単位として扱う場合，ブロックサイズは，ファイルシステムの種類や設定によって異なるために，本システム利用者が適切なブロックサイズを設定する必要があるが，OS が行う入出力の単位で処理できるため，速度と容量の両面で効率がよい．セクタを単位として扱う場合，ブロック単位よりも細かい単位で処理することになるため，効率は落ちるものの，ファイルシステムに依存せずに処理できる利点がある．性能改善のためのオプションとして，提案システムでは書き込みの単位を可変にできるものとして設計，実装した．

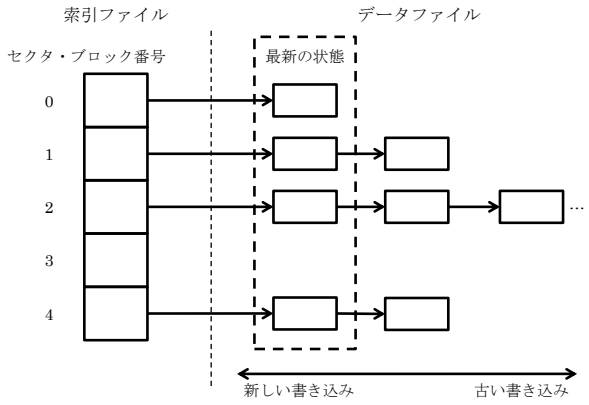

図 4 データ構造

データ構造は，(1) リストのエントリを格納するデータファイル，(2) データファイルを探索するための索引ファイル，の 2 つとして保管される．データファイルに格納されるエントリは，図 5 に示すように書き込みの行われた時刻，リストの次のエントリの位置，書き込み内容のサイズ，書き込み内容本体の 4 項目からなる．時刻は，書き込みが行われたナノ秒単位での時刻を示す．書き込みのサイズは書き込み内容のサイズ，書き込み内容は補助記憶装置への書き込むデータをバイト単位で示す．OS のファイルシステムはブロック単位で書き込みを行うため，書き込みのサイズは常に一定であるが，常にブロック全体が意味のあるデータになっているとは限らず，データが存在しない末尾が 0 で埋められている場合がある．ブロックによっては先頭の数バイトを除いてすべてが 0 になる場合もあり，それらをそのまま格納するとデータファイルが肥大化するため，末尾の 0 の連続した部分はエントリに含めず，代わりに末尾を除いた部分のサイズを格納しておき，読み込み時に末尾の 0 の並びを復元する．

次のエントリの位置は，ファイルの先頭からのバイト単位でのオフセットで表現する．ただし，0 は次のエントリ

が存在しないことを示す特別な値として利用する．次のエントリの位置を 32 ビットの整数値とすると，512 バイト単位で処理する場合で約 2TB，4096 バイト単位で処理する場合で約 16TB 分のデータを保存すると上限に達してしまうため，将来的に補助記憶装置の容量が拡大することを想定して 64 ビットとしている．

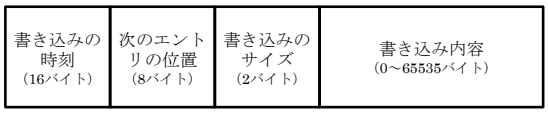

図 5 エントリの内容

索引ファイルは，各セクタ・ブロックのリストの先頭エントリ，すなわち最新のエントリのデータファイル上でのバイト単位の位置を並べたファイルである．索引ファイルからセクタ・ブロックの先頭のエントリの位置を取得し，データファイルの該当部分を参照することで，各セクタ・ブロックの最新の状態を取得でき，リストを辿ることで時系列順に過去のデータを探索できる．リストなので追記するときもデータファイルへ追記していくだけで良いという利点がある．なお，高速化のために索引ファイルの内容はあらかじめメモリ上に展開しておき，読み込みのみの場合には索引ファイルにアクセスしない．

6.2 保全時の処理

書き込みの保全を行う tapdisk ドライバは，本システム利用者に対しては通常の補助記憶装置と同様に振る舞いながら，6.1 節に示したデータ構造に基づいて読み書きを行う．OS から読み込み要求を受けた場合の処理の手順を図 6 に示す．対象セクタ・ブロックの索引から 0 以外の有効なエントリの位置が取得された場合，そのエントリがそのセクタ・ブロックの最新のエントリであるため，最新のエントリからデータを読み込んで返す．索引から 0 が取得された場合，一度も書き込みが行われていないため，すべて 0 のバイト列を返す．

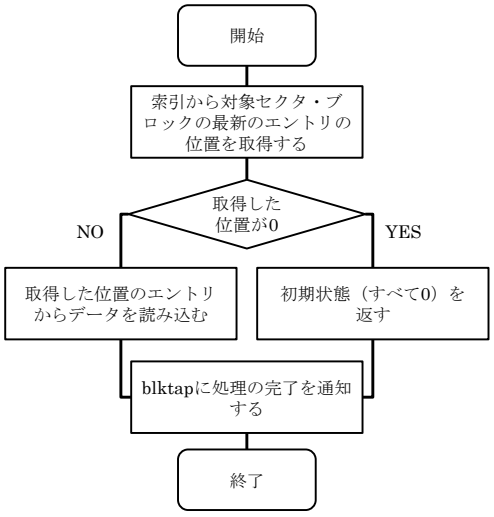

図 6 読み込み処理のフローチャート

OS から書き込み要求を受けた場合の処理のフローチャートを図 7 に，リストの変化の様子を図 8 に示す．データファイルと索引ファイルで構成されるリスト構造は，対象セクタ・ブロックへの書き込みが新しいものから順に並ぶようにする必要があるため，新たに書き込みが行われた場合は，エントリを作成してリストの先頭に挿入する．作成したエントリは，データファイル末尾に追記する．これにより，書き込みが行われるのはデータファイルの末尾と索引ファイルの該当セクタ・ブロック部分のみとなりデータファイルの既存のエントリに変更を加える必要がなくなる．

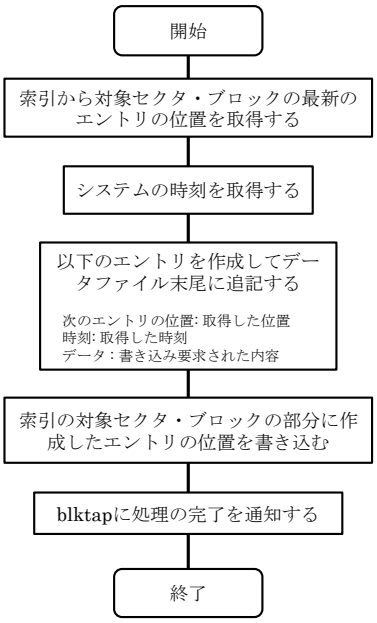

図 7 書き込み処理のフローチャート

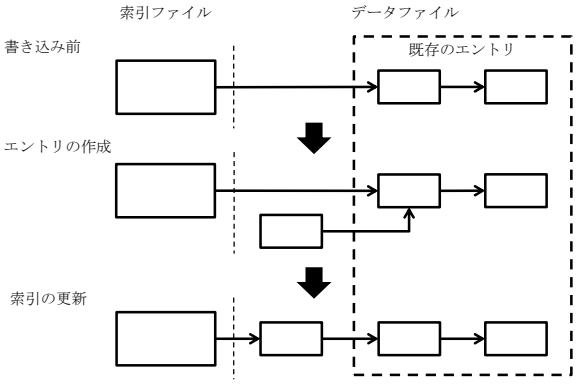

図 8 書き込み時のリスト操作

6.3 復元時の処理

状態の復元を行う tapdisk ドライバは，指定された時刻の状態を再現した読み込み専用の補助記憶装置として振る舞う．OS から読み込み要求を受けた場合の処理を図 9 に示す．復元する時刻の状態とは，指定された時刻より古いエントリの中で最新の状態を指しているので，対象セクタ・ブロックのリストを最新のエントリから順に辿り，復元する時刻より古いエントリが見つかり次第探索を打ち切ることで発見できる．リストを末尾まで辿っても復元する時刻

より古いエントリが存在しない場合，その時刻の時点では初期状態から変化していないことを意味するため，すべて 0 であることを意味する．

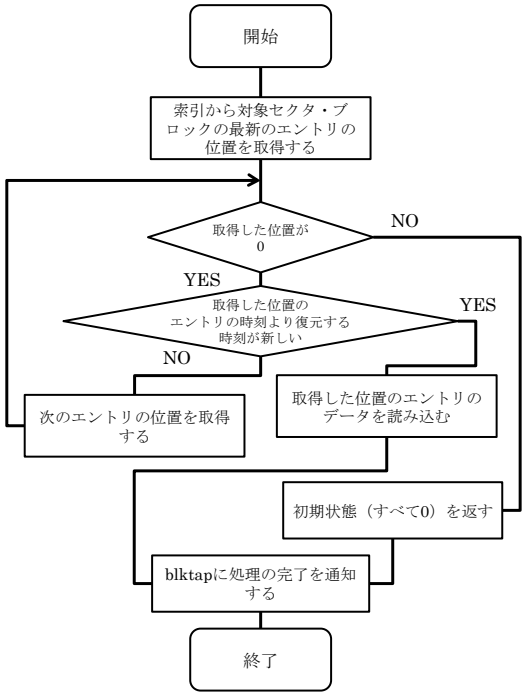

図 9 状態を復元する処理のフローチャート

状態の復元時にデータファイルおよび索引ファイルの状態が変化することはないため，実際にはエントリを探索する処理は最初に参照された時に一度だけ行い，二度目以降は一度目の探索結果をキャッシュして利用することで高速化を行う．また，データファイルと索引ファイルに対して書き込みを行わないため，複数のプロセスから同時にアクセスしても問題が生じないので，同時に異なる時刻の複数の状態を復元しても不整合が生じず，差分を比較できる．

7. 運用方法

提案システムを利用する場合に考えられる運用方法について述べる．まず，提案システムは仮想計算機モニタのレベルで動作することから，テナントに対してサービスを提供するクラウドサービスプロバイダ（CSP）によって運用されることを仮定する．この場合，テナントは履歴付きのフォレンジックサービスを必要なドライブに対してオプションで追加できるものとする．CSP は管理用のドメイン 0 へのアクセスはテナントには認めず，論理ネットワーク的にも切り離して運用するものとする．また，履歴データの保管先の物理ディスクには他のサービスと共有している分散ストレージを利用しないことを提案する．以下では CSP でログファイルの保全を行う場合の運用例を説明する．

7.1 保全フェイズ

ログファイルの保全は，保全用の tapdisk ドライバを用いた仮想ディスクを，指定したドメイン U からアクセスできるよう設定することで実現する．本稿では保全用の tapdisk ドライバは，OS がインストールされたディスクイメージに対しては適用せず，アクセスログを記録するためのディスクイメージを別途作成し，その仮想ディスクに対してのみ保全機能を適用することを提案する．ドメイン U からは，新しいハードウェアが追加されたように見えるので，仮想ディスクをマウントし，アプリケーションやログ出力デーモンの出力先として指定する．

7.2 分析フェイズ

保全した仮想ディスクの分析は，指定した時刻の状態を復元する tapdisk ドライバを用いて行う．blktap には，ドメイン U を利用せず，ドメイン 0 で tapdisk ドライバの機能を利用できる仕組みが用意されているため，分析は主にドメイン 0 の環境で行う．Xen と blktap が導入されている環境であれば，保全対象とは異なるマシン上で解析を行うことも可能である．

tapdisk ドライバを用いてマウントした仮想ディスクは，利用する OS からは単に新しいハードディスクがマウントされたように見えるため，既存の解析ツールや Linux のシステムツールを活用して分析できる．複数の tapdisk ドライバを同時に利用することもできるため，複数の時刻の状態を同時に復元し，それぞれ別な仮想ディスクとしてマウントし，既存のツールを用いて比較することで，ファイルシステムに行われた改竄や削除の履歴を特定できる．以降では，分析手法として，diff コマンドを利用する方法と，時系列順でファイルを比較する方法について述べる．

7.2.1 diff コマンドによる分析

diff コマンドを利用して分析する例を図 10 に示す．例では 2 つの時刻の状態を再現する tapdisk ドライバを起動し，それぞれを読み取り専用でマウントして Linux の diff コマンドによってファイルシステムの状態を比較している．diff コマンドに-R オプションを付けて実行すると，指定された 2 つのディレクトリ内を比較し，片方にしか存在しないファイルや内容が異なるファイルを出力する．

```
disk_path=/path/to/disk
date1=`date +%s –d ‘2013-01-01 00:00:00’`
date2=`date +%s –d ‘2013-01-02 00:00:00’`
disk1=`tap-ctl create –a analysis:$disk_path:$date1`
disk2=`tap-ctl create –a analysis:$disk_path:$date2`
mount –t ext4 –o ro,noload $disk1 /mnt/disk1
mount –t ext4 –o ro,noload $disk2 /mnt/disk2
diff –R /mnt/disk1 /mnt/disk2
```

図 10 diff コマンドを利用する例

7.2.2 時系列順での比較による分析

時系列順でファイルを比較することで，改竄や削除の時刻を検出することができる．このためには，一定間隔で複数の時刻でのログファイルの状態を復元し，隣り合う時刻

の状態を比較する．時系列順で比較する方法を図 11 に示す．ログファイルに対しては，末尾への追記のみが行われるため，末尾以外の部分に変更点が存在した場合は改竄・削除が行われたものとみなすことができる．本研究では上記方法でログファイルの改竄時刻を検出する Ruby スクリプトを実装し動作確認を行った．

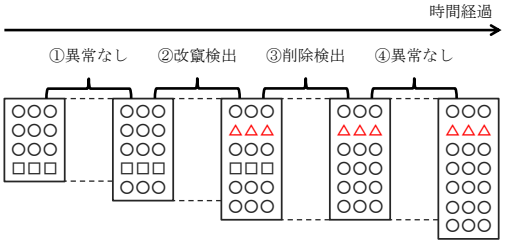

図 11 時系列順での比較

8. システム評価

提案システムで補助記憶装置に読み書きされた入出力データを保全した際，必要となる補助記憶装置の容量の増加を計測した．続いて，保全用の tapdisk ドライバを利用してドメイン U のゲスト OS が補助記憶装置へ読み書きした際の速度を測定した．評価環境を表 1 に示す．なお，7.1 節で述べたように，測定は OS がインストールされた仮想ディスクとは別のディスクに対して適用して実施した．メモリ割り当ては，ドメイン 0 に 3.5GiB，ドメイン U に 512MiB とした．

表 1 評価環境

機種	DELL Inspiron 570
CPU	AMD Athlon II X2 Dual-Core 245 2900MHz
メモリ	DDR3 SDRAM 4GiB
ハードディスク	Serial ATA 500GB（7200rpm）
仮想計算機モニタ	Xen 4.1.2（blktap2）
OS（ドメイン 0）	CentOS 6.2（64bit）
OS（ドメイン U）	CentOS 5.8（32bit）
ファイルシステム	ext4

8.1 保存に必要な補助記憶装置の容量

提案システムを利用してゲスト OS で単一のファイルを作成した場合に，実際に使用される補助記憶装置の容量の増加を測定した．ファイル作成は，フォーマット直後のディスクイメージを保全用の tapdisk ドライバを用いてマウントし，マウントしたディレクトリに別のディレクトリからファイルをコピーして行った．その後，ディスクをアンマウントし，ファイルの容量を測定した．なお，索引ファイルのサイズは変化しないため，データファイルのサイズのみを測定している．作成するファイルのサイズを 100KiB，1MiB，10MiB，100MiB，1GiB と変化させて測定を行った結果を図 12 に示す．図 12 は，作成したファイルサイズに対して，本システムのデータファイルの利用量が何倍になっているかを示したものである．

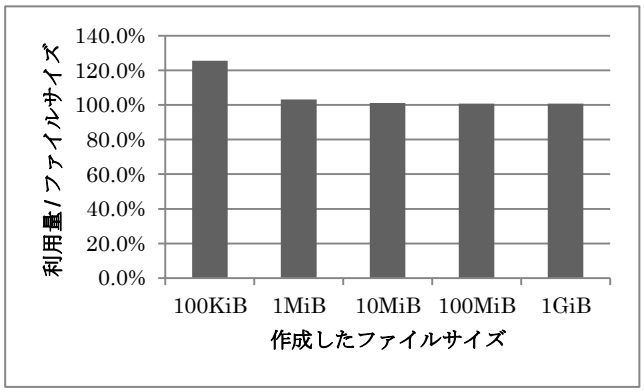

図 12 ファイルサイズと補助記憶装置の利用量

8.2 読み書きの速度

補助記憶装置の入出力ベンチマークソフトウェアである bonnie++ 1.03e[8]を利用し，保全機能を組み込んでいない標準の tapdisk ドライバ（aio）を利用する場合と，保全機能を組み込んだ tapdisk ドライバを利用する場合の入出力の速度を比較した．入出力速度の測定結果を図 13 に示す．Char は文字単位，Block はブロック単位での読み書きを評価した結果である．

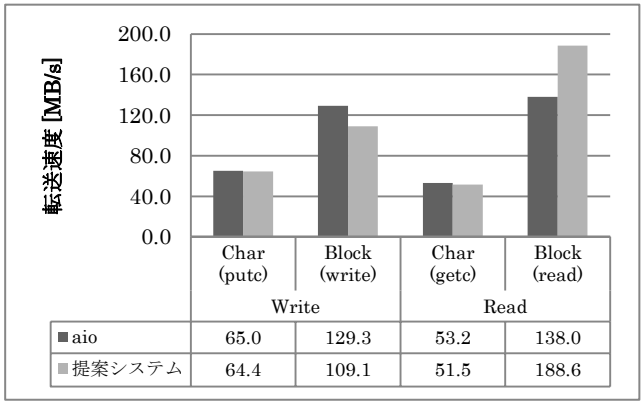

図 13 入出力速度の測定結果

9. 考察

図 12 より，100KiB のファイルを作成した場合にはファイルサイズの 1.25 倍の容量を使用している一方，1MiB の場合は 1.03 倍になっている．これは，ファイルシステムが管理するファイルの情報に，ファイルサイズに関係なく一定の容量を必要とするためと考えられる．時刻とリスト構造を保存するために，作成するファイルサイズに応じた増加量が生じているものの，この増加量は，図 5 よりセクタ・ブロック 1 つにつき 26 バイトなので，ext4 をデフォルトの設定で利用する場合には 1 ブロック（4096 バイト）あたり約 0.63%の増加である．扱うファイルサイズが小さい場合にはファイルシステムの情報による増加量が問題になるが，本稿で提案システムが対象としているログファイルは，ローテートを行わない限り 1 つのファイルに追記し続けられるため，細かなファイルが多数生成されることはな

い．よって，ログファイルを扱う上では本システムのファイルサイズの増加量は致命的な問題にはならないと考えられる．

図 13 より，提案システムを有効にすることによってブロック単位の書き込みで約 20%低下することが確認できた．逆に読み込みでは約 50%転送速度が向上していることを確認した．書き込みはデータファイル末尾と索引ファイルの連続した領域に行われるため，本システムの適用による速度低下が最小限に抑えられている．読み込みは必ずしも連続した領域に対して行われるわけではないためデータファイルに対するシークが発生する．このため読み込み速度も低下するはずであるが，今回の実験では性能が逆に向上している．この理由として考えられるのはドメイン 0 のファイルシステムのキャッシュ機能である．標準の tapdisk ドライバ（aio）はファイルを開く際に O_DIRECT オプションを付けて実行されるためファイルシステムのキャッシュが無効化されている．しかし，提案システムの実装では O_DIRECT オプションを付けて実行することができないため，キャッシュによる高速化の影響が表れていると考えられる．提案システムが想定するログファイルは，普段は読み込みを行うことはほとんどなく，書き込みが頻繁に行われる．毎秒 109.1MB に落ち込む文字単位での書き込みであっても，24 時間書き込み続けたと仮定した場合に約 9.4TB のデータを書き込むことができる．

提案システムは，OS やファイルシステムに依存しないセクタ・ブロック単位での処理を実現するが，保全したデータの分析時には，対象となるデータのファイルシステムを解釈できるソフトウェアが必要である．これらの解析用ソフトウェアには商用の EnCase や Linux でも使える The Sleuth Kit が挙げられる．他方，暗号化されたファイルシステムをそのままの状態で分析することはできないため，分析するためには鍵を入手するか，解析ソフトウェアの解読機能を用いる必要がある．

本提案は補助記憶装置をコンピュータから取り外して外部で行われる改竄には対応できない．このため適用先をクライアントコンピュータではなく CSP が運用するサーバシステムとしている．対策としては鶴田らの研究[2]で行われているように，保存するデータに対してデジタル署名を行い，改竄を検出できるようにすることが考えられる．

本稿で述べた提案システムは仮想計算機モニタの Xen によって実装している．Xen の準仮想化は性能面で優位性があるため CSP でも利用されている方式である．このように仮想計算機モニタ上にセキュリティ機能を実装することを提案したが，仮想計算機モニタ自体が改竄されてしまった場合にセキュリティ機能が無効化される危険性は排除できない．仮想計算機モニタの改竄への一つの対策としてトラステッドブートを利用する方法がある．トラステッドブートはソフトウェア実行開始時にソフトウェアの改竄を検出する仕組みであり，本システムへの適用が考えられる[9]．セキュリティ機能を組み込んだ仮想計算機モニタ自体の改竄を試みる攻撃者がいてもトラステッドブートを利用して仮想計算機モニタを含む提案システム全体（ドメイン 0 のカーネルを含む）の完全性を事前に検査することはできる．しかしながら仮想計算機モニタと提案システムが実行中に攻撃された場合にはトラステッドブートでは対応できないため，実行中の仮想計算機モニタを保護するための別な対策が必要である．この防御機構の必要性は従来の OS に対するものと同様である．

10. おわりに

本稿ではクラウドコンピューティング環境で利用されている仮想計算機モニタを利用した仮想ストレージの保全と分析のためのシステムを提案した．関連研究では攻撃者によってファイルを改竄されることを防ぐための対策を列挙した．本稿では仮想計算機モニタのデバイスドライバを利用して，仮想ディスクへの書き込みを全て時系列に保存するシステムの設計と実装を示した．本稿では実装したシステムを用いてアクセスログの改竄時刻を検出する方法についても示した．続いて実装したシステムにおけるデータファイルの利用量の増加と転送速度について性能評価を行った結果を示した．最後に提案システムの解決すべき課題について述べた．

謝辞 本研究は JSPS 科研費 23700095 の助成を受けたものです．

参考文献

1) Karen Kent, Suzanne Chevalier, Tim Grance, Hung Dang: Guide to Integrating Forensic Techniques into Incident Response, National Institute of Standards and Technology Special Publication 800-86, pp.121 (2006).
2) 鶴田浩史, 齋藤彰一, 上原哲太郎, 松尾啓志: ヒステリシス署名による改竄防止機構を備えたログ用ファイルシステムの提案, 情報処理学会研究報告 CSEC, pp.235-240 (2009).
3) 高田哲司, 小池英樹: 逃げログ: 削除まで考慮にいれたログ情報保護手法, 情報処理学会学会誌, Vol.41 No.3, pp.823-831 (2000).
4) 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, Vol.1, No.2, pp.275-284 (2008).
5) 忠鉢洋輔, 品川高廣, 加藤和彦: 仮想マシンモニタによるゲスト OS のファイル保護: 情報処理学会研究報告 計算機アーキテクチャ研究会報告, Vol.2009-ARC-183, No.11, pp.1-8 (2009).
6) Joel Young, Kristina Foster, Simson Garfinkel, Kevin Fairbanks: Distinct Sector Hashes for Target File Detection, IEEE Computer, Vol.45, No.12, pp.28-35 (2012).
7) Citrix Systems: Blktap – Xen, http://wiki.xen.org/wiki/Blktap
8) Bonnie++, http://www.coker.com.au/bonnie++/
9) 電子情報技術産業協会: 「セキュア・プラットフォームに関する技術動向」調査報告書 (2008).