Vol.2012-CSEC-59 No.6
2012/12/14

ライブラリの置き換えによるVM外部への安全なログ転送方式の提案

佐藤 将也[1,a)]　山内 利宏[1]

概要：ログは，計算機の動作を把握するための重要な情報である．しかし，攻撃や問題の発生により，ログの改ざんや消失が起こる可能性がある．この問題へ対処するために，ログを保護する手法が提案されている．しかし，手法の多くは AP や OS で実現されており，カーネルレベルで動作するマルウェアからログを保護するのは難しい．そこで，我々は，VMM を用いたログの保護方式を提案した．しかし，この方式では，複数の VM において多種の OS が利用される場合，それぞれの OS に対応するように VMM を修正する必要があり，その工数が大きい．そこで，複数 VM 上の多種の OS に最小限のプログラムの修正により対応可能な，AP の出力するログの保護方式を提案する．提案方式は，ログ発行時に特定の命令を実行するように，VM 上のライブラリをあらかじめ置き換える．VMM は，この命令を契機にログを保護する．このため，提案方式は OS の種類に依存しない．

1. はじめに

計算機の動作を把握するためには，計算機上のログが重要である．しかし，攻撃や問題の発生により，ログの改ざんや消失が起こる可能性がある．この問題へ対処するため，我々は，仮想計算機モニタ（以降，VMM）を用いてログの改ざんや消失を防止する方式を提案した [1]．この方式では，仮想計算機（以降，VM）上で動作する応用プログラム (以降，AP) とオペレーティングシステム（以降，OS）のログを，AP や OS の修正なしに VMM が取得する．また，取得したログは，保護対象の VM とは異なる VM で保護する．このため，取得したログの安全性を確保できる．

この方式の利点は，VM 上の AP や OS を修正する必要がない点である．VM を用いる環境では，1 台の計算機上で複数の VM が動作し，それぞれの VM 上では多種の OS が動作することが想定される．しかし，既存手法 [1] では，複数 VM 上で多種の OS が動作する環境に対応するには，VMM が各種の OS に対応する必要がある．対応のためには，VMM は，複数の VM 上の AP の状態を監視する必要がある．また，多種の OS がログを送信する際のシステムコールの呼び出し順を事前に把握しておく必要がある．このように，複数 VM 上で多種の OS が動作する環境に対応するには，VMM が膨大な情報を管理する必要がある．また，OS のバージョンアップなど，OS ごとの修正にも対応する必要がある．

さらに，この方式では，ログを転送するシステムコールを監視するために，VM 上で発行されるすべてのシステムコールを検知する．この際，処理が VM から VMM へ遷移する．この遷移におけるオーバヘッドにより，VM 上の AP の性能が低下する．また，複数の VM が動作する環境では，他の VM の性能も低下する恐れがある．

本稿では，これらの問題を解決するために，VM 上のライブラリを置き換えることで，VMM を OS ごとに改変する必要なしに多種の OS に対応する方式を提案する．提案方式では，ログ発行時に特定の命令を実行するように，VM 上のライブラリをあらかじめ置き換える．VMM は，この命令を契機にログを VM から取得する．このため，VMM は，OS の種類への依存なしに VM 上で発行されたログを取得できる．VMM は，取得したログをログ保存用の VM に転送する．これにより，多種の OS への適用が容易になる．さらに，VM 上で発行されるすべてのシステムコールを検知する必要がなく，ログの転送時に実行される命令のみ検知する．このため，不要なオーバヘッドが発生しない．

本稿では，VMM として Xen を用い，VM 上で FreeBSD を動作させた場合において提案方式を実現し，その有効性を評価した．

2. 研究背景

2.1 Linux 上の AP におけるログの保存経路

Linux 上の AP におけるログの保存経路と経路上で想

[1] 岡山大学 大学院自然科学研究科
[a)] m-sato@swlab.cs.okayama-u.ac.jp

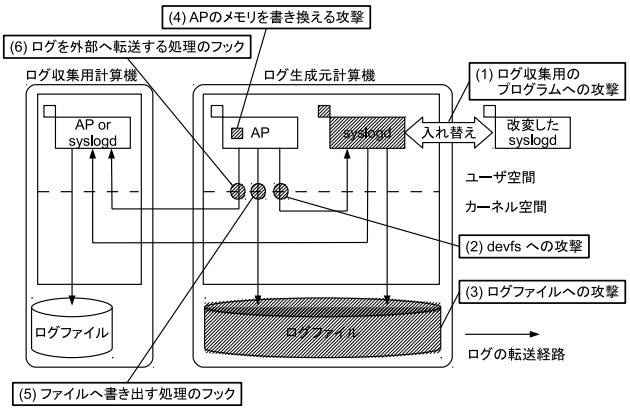

図 1　想定する攻撃箇所

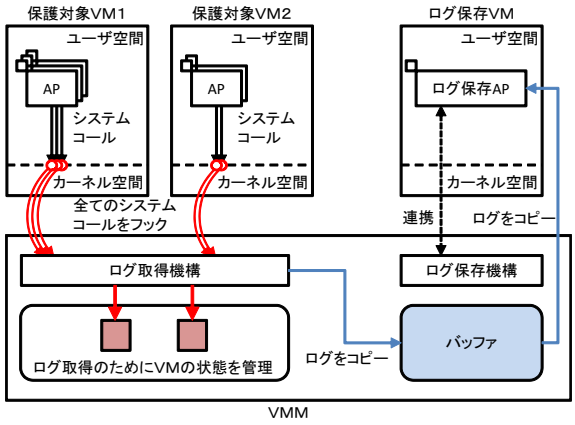

図 2　仮想計算機モニタによりログの改ざんや消失を防止するシステム [1] の全体像

定される攻撃を図 1 に示す．図 1 のように，ログの保存方法は 2 通りある．ログ収集用のプログラム（図 1 中の syslogd）を利用する方法と直接出力する方法である．

いずれの方法においても，ログがファイルへ書き出されるまでにカーネルを経由する．このため，ログ生成元計算機がカーネル内で動作するマルウェアに感染した場合，カーネルを経由する際にログが改ざんされる恐れがある．

2.2　想定する攻撃

図 1 に示すように，Linux 上の AP におけるログの保存経路上では，以下の攻撃が想定できる．

(1) ログ収集用のプログラムを置き換える攻撃
(2) devfs への攻撃
(3) ログファイルへの攻撃
(4) AP のメモリ領域を書き換える攻撃
(5) ファイルへ書き出す処理のフック
(6) ログを外部へ転送する処理のフック

(1) の攻撃を行うマルウェアとして，tuxkit[2] がある．tuxkit は，ログ収集用プログラムである syslogd を，改変した syslogd と置き換える．改変した syslogd は，悪意ある活動に関するログをファイルへ書き出さない．

(2) の攻撃を行うマルウェアとして，adore-ng[3] がある．adore-ng は，通信に利用されるカーネル内の関数を改ざんし，悪意ある活動に関するログの転送を阻止する．

(3) の攻撃は，ログファイルの書き込みや読み込みの権限を攻撃者が持つ場合に可能である．

(4) の攻撃が可能なマルウェアとして，LKM ルートキットがある．LKM ルートキットは，カーネル空間で動作する．このため，ユーザ空間で動作する AP のメモリ領域を書き換え，AP の生成したログを改ざんできる．

カーネルレベルで動作するマルウェアは，プロセス間通信やファイルアクセスなど，カーネルへの処理依頼を必要とする処理を攻撃できる．このため，(5) や (6) のように，カーネルを経由する処理は，攻撃される危険がある．

2.3　仮想計算機モニタによりログの改ざんや消失を防止する手法

我々はこれまでに，2.2 節で述べた攻撃へ対処するために，仮想計算機モニタによりログの改ざんや消失を防止する手法 [1] を提案した．この手法の全体像を図 2 に示し，概要を以下で説明する．

この手法では，ログを保護する対象の計算機として VM を想定する．AP の出力するログ（以降，ユーザログ）の出力では，システムコールが発行される．このため，ユーザログの取得には，VM 上で発行されるシステムコールの VMM による検知を用いる．カーネルの出力するログ（以降，カーネルログ）は，カーネル内の関数により出力される．このため，カーネルログの取得には，VM 上で発生するブレークポイント例外の VMM による検知を用いる．VM 上に OS がロードされた直後に，VMM が VM のメモリを操作し，カーネルログを出力する関数内にブレークポイントを埋め込む．これを契機とし，VMM は VM 上のカーネルログを取得する．

この手法により，2.2 節の攻撃のうち，(4) 以外の攻撃へ対処できる．しかし，この手法には，以下の問題がある．

(1) ログの取得による性能低下
(2) 多数の VM 上で多種の OS が動作する環境への適応の難しさ

この手法は，VM 上で発行されるシステムコールをもとに，どのプロセスがいつ syslog デーモンにログを送信しようとしているのかを判断している．このため，この手法は VM 上で発行されるすべてのシステムコールの発行を監視する．システムコールの監視では，VM 上でシステムコールが発行された際に，VM exit を起こすように VM のレジスタを VMM により書き換えている．このため，すべてのシステムコール発行において，VM から VMM へ処理が遷移する．VM から VMM への処理の遷移にはオーバヘッドが発生するため，VM 上の多くの処理の性能が低下する．

また，この手法は，上述したように，VM 上のシステム

コールの発行を監視し，ログの送信を判断している．このため，VMM は，ログの送信にどのようなシステムコールがどのような順序で発行されるかをすべて事前に把握しておく必要がある．また，複数の VM が動作するような環境では，すべての VM が同じ OS を搭載しているとは限らず，多種の OS が混在する環境が想定できる．このような環境に対応するためには，VMM は，各 VM ごとに多種の OS に対応した情報をすべて管理する必要があり，管理コストが大きい．

3. 脅威モデル

3.1 想定する脅威

本研究では，以下の攻撃を想定する．

(1) カーネル内に攻撃コードを挿入する攻撃
カーネル内に攻撃コードを挿入する攻撃を想定する．このため，本研究では，カーネルは信頼できないものとする．カーネル内に攻撃コードを挿入可能な場合，攻撃者は，ログの送信やファイルへの書き出しの際に発行されるシステムコールなど，ログに関する処理がカーネルモードに遷移した際に，ログを改ざんできる．

(2) ログ収集用プログラムの実行ファイルの置き換えやメモリの書き換え
ログ収集用のプログラムの実行ファイルを，ログを改ざんするように修正したプログラムと置き換える攻撃を想定する．攻撃者がプログラムを置き換えた場合，利用者にとって必要なログが改ざんされる．また，ログ生成元のプログラムやログ収集用プログラムがメモリにロードされた後に，ロードされたプログラムのメモリ領域を書き換えるような攻撃を想定する．

(1) のようにカーネル内に攻撃コードを挿入された場合は，ユーザ空間のメモリを書き換え可能である．ユーザ空間のメモリが書き換えられた場合，提案手法によるライブラリ置き換えの効果がなくなる．このため，この攻撃に対しては，文献 [4] の手法により AP のメモリ空間を保護できるものとする．

LKM を無効化することで LKM による攻撃を回避できる．しかし，カーネルの柔軟性を大きく損なうため，本研究では LKM の無効化は想定しない．また，/dev/kmem などを用いたカーネル内のデータを書き換える攻撃を想定する．これらの攻撃のように，ログやログの出力に関するプログラムへの攻撃を想定する．

本研究では，Linux や FreeBSD などの UNIX 系 OS だけでなく，Windows などのオープンソースではない OS における脅威も想定する．しかし，提案方式は，ライブラリを置き換える必要があるため，Windows における実現は困難である．このため，Windows においては，DLL インジェクションにより，ライブラリの置き換えと同等の機能を実現する．この詳細については，7.2 節で述べる．

3.2 想定しない脅威

ログ生成元のプログラム自体が動作しないようにする攻撃は想定しない．例えば，カーネルレベルで動作するマルウェアを用い，特定のプロセスがスケジューリングされないようにランキューを操作するような攻撃は想定しない．このような攻撃には，SecVisor[5] により対処する．

また，VMM を対象とした攻撃は困難であることを前提とする．VMM を攻撃する手法 [6][7] が報告されているが，いずれの攻撃も特定の条件においてのみ可能であり，これらの攻撃は回避できるものとする．

4. ライブラリの置き換えによる VM 外部への安全なログ転送方式

4.1 目的と要件

3.1 節で述べた脅威によるログの改ざんを防止するには，カーネルにログが到達する前に保護すればよい．また，保護する機構や保護したログを攻撃から保護するには，VMM の利用が有効である．

このため，本研究では，VM 上の AP からカーネルを経由せずに VM 外部へ安全にログを転送することを目的とする．また，2.3 節で示した既存手法 [1] の問題を解決する．

この目的を実現するためには，以下の要件がある．

（要件 1） VM 上の AP の出力したログがカーネルを経由しないこと

（要件 2） OS の種類やバージョンの違いに容易に対応できること

（要望 1） ログの転送におけるオーバヘッドを抑制すること

（要件 1）は，カーネルレベルで動作するマルウェアによるログの改ざんを防止するために必要である．また，OS の種類やバージョンの違いに容易に対応することが重要である．さらに，ログの転送の際に発生するオーバヘッドを最小限に抑えることが望ましい．

4.2 課題

4.1 節で示した要求を満たすための課題を以下に示す．

（課題 1） カーネル到達前のログの VM 外部への転送

（課題 2） 既存手法 [1] における不要な VM exit の削減

（課題 3） 最小限のプログラムの修正による課題の実現

（要件 1）を満たすために（課題 1）がある．ただし，（課題 1）は，既存手法 [1] によりすでに実現しているため，この課題の解決には，既存手法 [1] の成果を用いる．（要望 1）を満たすために（課題 2 ）がある．また，これらの課題を最小限のプログラムの修正により解決することで，（要件 2）で示した要求を実現する．

4.3 提案方式

図 3 に提案方式の全体像を示す．提案方式は，VMM を

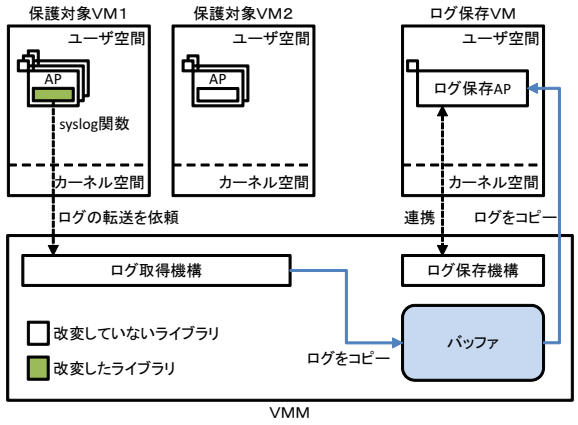

図 3　提案方式の全体像

用い，ログの保護対象となる OS を VM 上で動作させる．以降，この VM を保護対象 VM とする．本稿で述べる VM は，VT-x を用い，完全仮想化されているものとする．提案方式では，保護対象 VM 上で出力されたログを，カーネルに到達する前に VMM が取得する．VMM は，取得したログを保護対象 VM とは異なる VM に転送する．以降，この VM をログ保存 VM とする．

提案方式では，保護対象 VM におけるログの出力を VMM により検知するために，保護対象 VM の AP が利用するライブラリを改変する．改変したライブラリは，ログを syslog デーモンに転送する処理の直前に，VM exit を起こす命令を実行する．これにより，VM から VMM へログの転送を依頼する．

提案方式では，以下の手順でログを VM 外部へ転送し，ファイルへ書き出す．

(1) AP は，ログの転送を VMM へ依頼
(2) VMM は，保護対象 VM 上の AP から VMM 内のバッファへログをコピー
(3) VMM は，ログ保存 AP へログをコピー
(4) ログ保存 AP は，ログファイルへログを書き出し

手順 (1) では，AP から直接 VMM を呼び出すため，VM 上のカーネルを経由しない．また，手順 (2) では，VMM が自身のバッファ内にログをコピーする．手順 (3) では，ログ保存 VM 上のログ保存 AP へログをコピーし，手順 (4) によりログ保存 AP がログをファイルへ書き出す．保護対象 VM のカーネルは，提案方式におけるログの保存経路上に介在しないため，（課題 1）へ対処している．

また，VM は，ファイルをディスクイメージに書き込む．ディスクイメージは VM ごとに分かれており，他 VM のディスクイメージへは，特権 VM 以外はアクセスできない．

さらに，提案方式では，既存手法 [1] と異なり，VM やプロセスの状態を VMM 内で管理する必要がない．このため，VMM 内の実装が簡易になる．また，VM 上で発行されるシステムコールをフックする必要がないため，不要な VM exit は起こらない．このことから，2.3 節で述べた手法に比べ，性能低下が小さいといえる．これにより，（課題 2）へ対処している．

（課題 3）への対処は，VM 上のライブラリの改変により実現する．（課題 3）への対処として，AP，ライブラリ，カーネル，および VMM の 4 か所での対処を検討した．AP による対処は，AP ごとにプログラムを修正する必要があり，修正工数が大きい．カーネルによる対処は，（課題 1）へ対処できず，（要件 1）を満たさない．VMM による対処は，OS の種類やバージョンの違いへの対応を VMM 内部で実現すると，VMM が複雑になり，バグが混入しやすくなる．一方，ライブラリによる対処は，AP よりも修正工数が小さく，カーネルへ到達する前にログを VM 外部へ転送できる．このため，ライブラリの改変により対処する．

表 1　実装に用いたソフトウェア環境

VMM	Xen 4.2.0
ゲスト OS	FreeBSD 9.0.0
Web サーバ	thttpd 2.25b
Web サーバベンチマーク	ApacheBench 2.3
マイクロベンチマーク	LMbench version 3

4.4　脅威への対処状況

提案方式では，AP が出力したログはカーネルを経由しないため，脅威 (1) へ対処できている．また，カーネルを経由せず，VM 外部へログを転送するため，ログ収集用プログラムを経由しない．このため，ログの保存経路上の他のプログラムからの攻撃によりログが改ざんされることはない．よって，脅威 (2) へ対処できている．

5.　実現方式

5.1　実装に用いたソフトウェア環境

表 1 に実装に用いたソフトウェア環境を示す．VM 上では，ゲスト OS として FreeBSD を用い，FreeBSD 上のライブラリを改変し，評価を行った．Linux と Windows においては，AP 内に CPUID 命令を実行する処理を挿入することで CPUID 命令を用いたログの転送依頼が可能なことを確認した．

5.2　AP から VMM へのログの転送依頼

AP から VMM へのログの転送依頼は，CPUID 命令をライブラリ内に埋め込むことで実現する．CPUID 命令は，CPU の情報を取得するための命令であり，CPUID 命令の実行により VM や CPU の状態が変化することはない．

VT-x を用いた VM 上では，レジスタの状態に関わらず VM exit を起こす命令が 16 種類ある [8]．この中から，VM の状態に影響を与えない CPUID 命令を VMM へのログの転送依頼の手段として用いる．

本稿における実現では，FreeBSD の libc 内の syslog 関

図 4　CPUID 命令を用いた VMM へのログの転送依頼

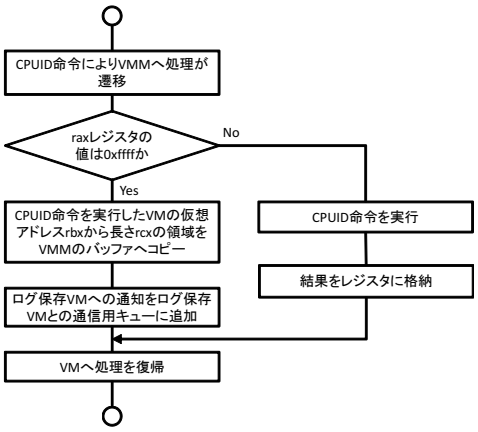

図 5　CPUID 命令による転送依頼の実行後の VMM によるログのコピー

数に CPUID 命令を実行する処理を追加した．syslog 関数内では，printf 関数のように，フォーマット文字列を受け取り，ログとして整形した後，syslog デーモンへログを転送する．このため，文字列をログに整形する処理の直後に CPUID 命令を実行する処理を追加した．

図 4 に CPUID 命令を用いた VMM へのログの転送依頼の処理の流れを示し，以下で説明する．

(1) AP は，rax レジスタに 0xffff を格納
(2) AP は，rbx レジスタにログを格納したバッファの先頭アドレスを格納
(3) AP は，rcx レジスタにログの長さを格納
(4) AP は，CPUID 命令により VMM へログの転送を依頼

CPUID 命令は，rax レジスタに格納された値に応じた CPU の情報をレジスタに格納する．文献 [8] では，CPUID 命令の際に rax レジスタに格納される値は，0x0 から 0xd，0x40000000 から 0x4fffffff，および 0x80000000 から 0x80000008 だと述べられている．このため，未使用の 0xffff を VMM へのログの転送依頼を示す値として用いた．

5.3　VMM による AP 上のログのコピー

図 5 に AP 上のログをコピーする処理の流れを示し，以下で説明する．

(1) CPUID 命令により VMM へ処理が遷移
(2) rax レジスタの値が 0xffff でない場合は (5) へ移動

```
Nov 13 03:41:23 debian logret: <38>Nov 13 03:41:22 sshd[1177]: Server listening on :: port 22.
Nov 13 03:41:23 debian logret: <38>Nov 13 03:41:22 sshd[1177]: Server listening on 0.0.0.0 port 22.
Nov 13 03:41:23 debian logret: <20>Nov 13 03:41:22 sm-mta[1183]: gethostbyaddr(172.21.23.236) failed: 1
Nov 13 03:41:23 debian logret: <22>Nov 13 03:41:22 sm-mta[1184]: starting daemon (8.14.5): SMTP+queueing@00:30:00
Nov 13 03:41:23 debian logret: <22>Nov 13 03:41:22 sm-msp-queue[1188]: starting daemon (8.14.5): queueing@00:30:00
```

図 6　提案方式により取得した VM 上の FreeBSD の起動ログ

(3) CPUID 命令を実行した VM の仮想アドレス rbx から長さ rcx の領域を VMM のバッファへコピー
(4) ログ保存 VM への通知をログ保存 VM との通信用キューに追加
(5) CPUID 命令を実行
(6) 結果をレジスタに格納
(7) VM へ処理を復帰

6.　評価

6.1　環境

VMM として Xen 4.2.0 を用い，ゲスト OS として FreeBSD 9.0.0 を用いた．提案方式を適用した計算機は，CPU は Core i7 2600（3.6 GHz, 4 コア，8 スレッド），メモリは 4GB 搭載している．なお，保護対象 VM には仮想 CPU を 2 コアとメモリを 1024MB 割り当て，ログ保存 VM には仮想 CPU8 コアとメモリ 1024MB を割り当てた．

また，6.5.4 項における Web サーバの性能測定では，提案方式を適用した VMM 上において，FreeBSD 上で Web サーバを動作させ，スループットを測定した．クライアント計算機は，提案方式を適用した計算機とは異なる計算機である．クライアント計算機は，CPU は Pentium 4（3.0 GHz，1 コア，1 スレッド），メモリを 1GB 搭載している．NIC は，提案方式を適用した計算機とクライアント計算機は 1Gbps 対応のものを搭載している．ただし，スイッチは 100Mbps のものを用いているため，測定時の通信路の帯域幅は 100Mbps である．

6.2　ログ取得実験

提案方式を導入した Xen 上の VM において，FreeBSD の起動ログを提案方式により取得した．ログ保存 AP は，VMM から受け取った保護対象 VM のログを，ログ保存 VM 上で動作する syslog デーモンを経由してログファイルに書き出す．取得したログの一部を図 6 に示す．

debian はログ保存 VM のホスト名であり，logret はログ保存 AP の名前である．図 6 から，保護対象 VM 上の AP が syslog デーモンへ送信したログを取得できていることがわかる．ログの先頭に付加されている〈38〉のような数字は，syslog デーモンへログを渡す時のパラメータである．syslog デーモンは，このパラメータと書き出しポリシをもとに書き出し先のファイルを決定する．syslog デーモンは，ファイルへ書き出す際にこの数字をログから取り除くため，ログファイルには記録されない．しかし，提案方

式では，AP が syslog 関数を呼び出した直後にログを取得している．このため，図 6 では，syslog デーモンによりパラメータが取り除かれる前のログが表示されている．

6.3 ログの保存経路における安全性

提案方式では，AP が syslog 関数を呼び出した際に VMM がログを取得する．VM から VMM への攻撃は困難であるため，VMM によるログ取得後の安全性はここでは検証しない．このため，AP がログを生成して VMM がログを取得するまでの期間における攻撃を検証する．

AP がログを生成した直後の攻撃を想定する．カーネルレベルで動作するマルウェアにより AP の動作や AP の保持するデータが書き換えられるような攻撃には，文献 [4] の手法により対処する．文献 [4] の手法では，ある AP の利用するユーザ空間のメモリをカーネルから書き換え不可能にするため，カーネルレベルで動作するマルウェアによるメモリ上のログの書き換えを防止できる．また，この手法を用いれば，ライブラリが展開されているメモリ領域への攻撃も防止できる．このことから，文献 [4] の手法と提案方式を組み合わせることで，あらゆる攻撃によるログの改ざんを防止できる．

次に，AP の実行ファイルを書き換える攻撃を想定する．この攻撃は，実行ファイルの完全性検証により防止できる．

以上から，提案方式と文献 [4] やファイルの完全性の検証手法を組み合わせた場合，ログを改ざんするような攻撃は，非常に困難であるといえる．

6.4 修正工数

修正工数として，プログラムを修正するために追加したプログラムの行数と修正箇所を決定するコストがある．

本稿におけるプロトタイプでは，FreeBSD 9.0.0 上の libc.so.7 内の syslog 関数に提案方式を適用した．追加した処理は，CPUID 命令を実行する処理と CPUID 命令の実行前にレジスタに適切な値を設定する処理である．追加したプログラムの行数は 20 行である．

修正箇所は，ログを送信する前に文字列を成型する処理の直後とした．UNIX 系 OS では，syslog 関数によりログを転送するため，対応する関数名をライブラリから探索することでライブラリの修正箇所を特定した．

ただし，実際に VM 上に適用する際は，利用者はもとのライブラリを修正を加えたライブラリに置き換えるだけでよい．このため，AP ごとに対応する必要はなく，提案方式の適用は容易だといえる．

6.5 提案方式の導入によるオーバヘッド

6.5.1 測定対象と測定の目的

提案方式の導入によるオーバヘッドを評価するために，以下の処理を行った際の実行時間を測定し，提案方式の導入前と導入後における性能を比較する．

(1) syslog 関数の呼び出し
(2) システムコール処理の呼び出し
(3) Web サーバ

修正を加えたライブラリ関数である syslog 関数の性能への影響を評価する．さらに，文献 [1] と比較して，提案方式ではシステムコール処理の呼び出しにおけるオーバヘッドが発生しないことを示すために，システムコール処理の呼び出しに必要な時間を測定する．最後に，AP の性能への影響を測定することで，実環境における利用を想定した評価を行う．

表 2 syslog 関数実行時の提案方式の導入におけるオーバヘッド（単位：μs）

	syslog
Xen	20.89
提案方式	63.62
オーバヘッド	42.72 (204%)

表 3 getppid システムコール発行時の提案方式の導入におけるオーバヘッド（単位：μs）

	処理時間
Xen	0.13
提案方式	0.13
オーバヘッド	0 (0%)

6.5.2 syslog 関数の呼び出し

表 2 に提案方式を導入した際の syslog 関数実行時におけるオーバヘッドを示す．表 2 より，提案方式の導入により，syslog 関数呼び出し時に 42.72 マイクロ秒のオーバヘッドが発生していることがわかる．

6.5.3 システムコール処理の呼び出し

提案方式は，文献 [1] の手法と異なり，VM 上で発行されるシステムコールをフックしない．このため，システムコール実行時に呼び出される共通処理（以降，システムコール共通処理）の性能への影響はない．このことを示すために，マイクロベンチマークソフトである LMbench を用い，システムコール共通処理の処理時間を測定した．LMbench は，システムコール共通処理の処理時間を測定するために，getppid を用いる．getppid は，カーネル内のプロセスディスクリプタから，システムコールを発行したプロセスの親プロセスのプロセス ID を取得するシステムコールである．このため，カーネル内での処理時間が非常に短く，システムコール共通処理の測定に適している．

表 3 にシステムコール共通処理の測定結果を示す．結果より，提案方式の導入による性能への影響はない．なお，文献 [1] の手法では，2 マイクロ秒のオーバヘッドが観測されている．

また，LMbench では，ファイル I/O やプロセスの生成と削除など，多くの項目について性能を測定するが，修正を加えていない Xen と提案方式を適用した場合では，大き

表 4　提案方式の適用による thttpd のスループットへの影響（単位：要求数/秒）

File Size	Xen	提案方式		
	平均	平均	最大	最小
1 KB	1121.94 (100%)	747.84 (67%)	1074.79	174.96
100 KB	61.89 (100%)	43.75 (71%)	75.13	16.69

な変化は観測できなかった．

6.5.4 Web サーバ

提案方式の導入による Web サーバの性能への影響を評価する．保護対象 VM 上で thttpd 2.25b を Web サーバとして動作させ，ApacheBench 2.3 により Web サーバのスループット（要求数/秒）を測定した．

測定では，ファイルサイズ 1KB と 100KB のコンテンツに対して，並列度 1，リクエスト数 100 に設定して連続で 5 回測定した際の平均値を用いた．**表 4** に評価結果を示す．

提案方式を導入した場合，導入していない Xen に対して，1KB ファイルでは 67% ，100KB ファイルでは 71%のスループットが得られた．

提案方式を適用していない Xen では，スループットに大きなばらつきはなく，安定した結果が得られた．一方，提案方式を導入した場合，スループットが大きく低下する場合があった．このため，表 4 には，スループットが最大の場合と最小の場合の測定結果を載せている．スループット低下の原因は，VMM からログ保存 VM へのログのコピーによるものだと推測できる．提案手法では，保護対象 VM から VMM へログをコピーした際に，一定量のログを VMM 内に蓄積する．その後，ログ保存 VM がスケジューリングされ，ログ保存 AP から要求を受け取ると，VMM からログ保存 AP へログをコピーする．このため，一定量のログが蓄積されるまでは，ログ保存 AP へのログのコピーは起こらない．

また，提案方式を適用していない Xen において，1KB ファイルを要求した際の応答時間は約 955 マイクロ秒である．syslog 関数実行時のオーバヘッドが約 42 マイクロ秒であることから，提案方式の適用による性能への影響は 4~5%程度だと推測できる．さらに，提案方式を適用していない Xen において，100KB ファイルを要求した際の応答時間は約 40.93 ミリ秒である．このことから，100KB ファイルを要求した場合の提案方式の適用による性能への影響は 0.1%程度だと推測できる．

測定結果からも，1KB ファイルを要求した際のスループットにおいて，提案方式を適用した場合の最大値は適用していない場合の平均値よりも一定して低い値が得られた．しかし，100KB ファイルを要求した場合は，提案方式を適用している場合の最大値が適用していない場合の最大値と同等か上回る場合があった．

これらのことから，VMM からログ保存 AP へのログのコピーが起こらない場合は，提案方式による性能低下はほとんど起こらない．特に，Web サーバの転送するデータサイズが大きくなるほど，提案方式が AP の性能へ与える影響は小さくなる．一方，VMM からログ保存 AP へのログのコピーが起こる場合は大きく性能が低下するといえる．

7. 考察

7.1 ログの転送依頼をするプロセスの認証

5.3 節で述べた手法では，ログを転送するプロセスを認証していない．このため，不正なプロセスが大量にログの転送依頼を発行し，計算機へ高負荷を与える攻撃が可能である．この攻撃では，ログの転送に VMM が関わることから，同一計算機上で動作するすべての VM の性能に影響を与える．このため，大量にログを送るような攻撃を防止する必要がある．

この問題への対処として以下の方法がある．

(1) ログの転送依頼を発行するプロセスを認証する方法
(2) 一定期間内に大量の転送依頼を発行するプロセスからのログの転送依頼を禁止する方法

(1) の方法では，認証により，不正なプロセスによるログの転送依頼を防止できる．しかし，どのプロセスを正常なプロセスとするかの判定基準を設ける必要がある．

(2) の方法では，不正なプロセスを転送依頼の発行頻度をもとに特定する．この方法では，ある期間におけるログの転送依頼の発行回数が一定数を超えた場合に不正なプロセスと判定する．この方法では，判定に用いる期間とログの転送依頼の発行回数の決定基準を検討する必要がある．

不正なプロセスとして判定した後に，以下の対処がある．

(1) ログの転送依頼の拒否
(2) プロセスの終了

ログの転送依頼を VMM が拒否した場合，一定時間経過後にログの転送依頼の発行回数が減少していた場合には，転送依頼の受付を再開するなど，柔軟に対処できる．しかし，すべてのプロセスによるログの転送依頼を管理する必要があり，管理コストが大きい．一方，プロセスを終了させる方法は，プロセスの終了後に何も管理する必要がないため，管理コストは小さい．しかし，誤って正常なプロセスを不正なプロセスと判断した場合に，重要なプロセスを終了させてしまう危険がある．

7.2 ライブラリの置き換えが困難な状況への対処

提案方式では，VMM による OS の種類やバージョンへの対処に VM 上のライブラリの置き換えを用いている．しかし，ライブラリの置き換えが困難な環境では実現が難しい．例えば，Windows では，特権ユーザであれば，ライブラリの置き換えは可能だが，ソースコードが公開されていないため，適切な場所に CPUID 命令を埋め込むことが難しい．

そこで，Windows では，DLL インジェクションの利用を検討している．DLL インジェクションにより，ログを転送する API を呼び出す処理の前にログの転送を要求する処理を埋め込むことで，提案方式を適用できる．

8. 関連研究

8.1 ログを保護する既存手法

図 1 で示した保存経路上の各段階において，ログの保護に利用できる既存研究について述べる．

(1) の攻撃を防止する手法として，syslog デーモンの完全性を検証する手法 [9] がある．この手法は，トラステッドブートと Late Launch により syslog デーモンの完全性を検証し，ログをリモート計算機へ転送する．

走行中のカーネルの完全性を保証する手法 [5] がある．この手法により，(2)，(5)，および (6) のようなカーネルへの攻撃を防止できる．この手法は，利用者が許可したコードのみをカーネル空間で実行可能にし，不正なコードの実行を防止する．しかし，利用者がどのコードの実行を許可するかの判断が難しい．

(3) の攻撃を防止するには，ログを読み込み専用のデバイスへ書き出す方法がある．また，NIGELOG[10] では，改ざんされたファイルを復元できる．

(4) の攻撃は，文献 [4] の手法により防止できる．文献 [4] の手法では，VM によるメモリアクセスを VMM が制御できる点に着目し，VM 上の特定の AP が利用するメモリ領域とその他の AP やカーネルが利用するメモリ領域を分割する．特定の AP が利用するメモリ領域は，カーネルを含む他のプログラムによる読み書きが禁止されている．これにより，(4) の攻撃を防止できる．しかし，この手法では，カーネルを経由するログの改ざんを防止できない．

8.2 ログを保護する既存手法の問題

8.1 節で述べた既存手法の問題を以下に示す．

(1) ログの保存において必ずカーネルを経由すること
(2) OS の種類やバージョンに合わせたカーネルの修正

既存手法では，ログが保存されるまでに必ずカーネルを経由する．カーネルがマルウェアに感染していると，カーネルを経由する際にログが改ざんされる恐れがある．

また，8.1 節で挙げた手法の多くは，OS として Linux を用いており，カーネル内に機能を実現している．しかし，Linux は，頻繁にカーネルが更新されている．通常，カーネルに修正を加えるのは困難な作業であり，既存手法をカーネルの更新に合わせて適用するのは修正コストが高い．

一方，提案方式では，VMM は，VM 上のログをカーネルへ到達する前に取得する．このため，(1) の問題は起こらない．また，提案方式は，OS の種類やバージョンへの対応は，VM 上のライブラリの修正により実現する．VM ごとのライブラリの修正は，FreeBSD において 20 行程度であり，修正工数は小さい．このため，多種の OS へ最小限の修正工数で対応できる．

9. おわりに

ライブラリの置き換えにより複数の VM 上で多種の OS が動作する環境に対応可能な，VM 外部への安全なログの転送方式を提案した．提案方式は，VM 上のライブラリを改変したライブラリと置き換えることで，VMM へのログの転送依頼を容易に実現できる．プロトタイプでは，VM 上の FreeBSD のライブラリへの変更は 20 行のみであった．

評価では，提案方式の導入によりログが安全に外部へ転送できることを実験により示した．また，性能への影響を評価した．Web サーバのスループットは，提案方式を導入した場合は，導入しない場合の 67%程度まで低下した．しかし，ログを VMM 内でバッファリングした場合は，性能への影響は非常に小さい．

今後の課題として Windows を対象とした提案方式の適用がある．また，多数の VM 上で多種の OS が動作する場面を想定した性能評価がある．

参考文献

[1] Sato, M. and Yamauchi, T.: VMM-Based Log-Tampering and Loss Detection Scheme, Journal of Internet Technology, Vol.13, No.4, pp.655–666 (2012).
[2] Analysis of a rootkit: Tuxkit, available from 〈http://www.ossec.net/doc/rootcheck/analysis-tuxkit.html〉
[3] A new Adore root kit, available from 〈http://lwn.net/Articles/75990/〉
[4] Dewan, P., Durham, D., Khosravi, H., Long, M. and Nagabhushan, G.: A hypervisor-based system for protecting software runtime memory and persistent storage, Proc. 2008 Spring simulation multiconference (SpringSim '08), pp.828–835 (2008).
[5] Seshadri, A., Luk, M., Qu, N. and Perrig, A.: SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes, Proc. 21st ACM SIGOPS Symposium on Operating Systems Principles, pp.335–350 (2007).
[6] Wojtczuk, R.: Subverting the Xen hypervisor, Black Hat USA (2008).
[7] Rutkowska, J. and Wojtczuk, R.: Preventing and Detecting Xen Hypervisor Subversions, Black Hat USA (2008).
[8] Intel: Intel 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C, available from 〈http://www.intel.com/Assets/PDF/manual/253669.pdf〉 (2009).
[9] Boeck, B., Huemer, D. and Tjoa, A.: Towards More Trustable Log Files for Digital Forensics by Means of "Trusted Computing", 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp.1020–1027 (2010).
[10] Takada, T. and Koike, H.: NIGELOG: Protecting Logging Information by Hiding Multiple Backups in Directories, International Workshop on Database and Expert Systems Applications, pp.874–878 (1999).