情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2011-OS-116 No.5
2011/1/24

IDS オフロードを考慮した仮想マシンへの動的メモリ割当

内田昂志[†1]　岡崎正剛[†1]　光来健一[†1,†2]

侵入検知システム（IDS）は不正なアクセスや侵入を検知することでシステムを保護しているが、最近ではまず IDS を攻撃して無力化することが増えてきた。仮想マシンを用いて IDS をオフロードすることにより IDS への攻撃を防ぐという手法が提案されている。IDS がオフロード先のメモリを使用するため公平性やメモリの利用効率の問題が生じる。　本稿では IDS のメモリ使用量を考慮して仮想マシンに動的なメモリ割り当てを行なう Balloon Performer を提案する。Balloon Performer は IDS のプロセスのメモリ消費量と IDS がファイルアクセスすることによって OS カーネル内に確保されるファイルキャッシュ量を定期的に測定する。そして、その分だけオフロード元の仮想マシンからオフロード先の仮想マシンへとメモリをを割り当て直すことで、IDS のメモリ使用量とオフロード元へのメモリ割り当ての合計を一定に保つ。我々は Balloon Performer を Xen に実装し、実験によりメモリ再割り当てが正確に行えていることを確認した。

Dynamic Memory Allocation to Virtual Machines on IDS Offloading

Takashi Uchida ,[†1] Seigo Okazaki [†1]
and Kenichi Kourai[†1,†2]

Although intrusion detection systems (IDSes) can detect illegal accesses, the attackers recently tend to disable IDSes before compromising the systems protected by them. IDS offloading with virtual machines (VMs) has been proposed, but there exist issues on fairness and efficiency because such IDSes use the memory of the VMs that they are offloaded to. This paper proposes Balloon Performer, which dynamically allocates memory to VMs considering memory consumption by IDSes. Balloon Performer measures the memory consumption of an IDS process and the size of file cache used by the process periodically. Then it reallocates that amount of memory from source to destination VMs so that the sum of the memory consumption by the IDS process and the memory allocated to the source VM is kept constant. We have implemented Balloon Performer based on Xen and experimentally confirmed that memory reallocation is accurate.

1. はじめに

システムへの不正アクセスを検出するために、侵入検知システム（IDS）が用いられている。IDS はシステムへの侵入を検知して、不正アクセスを管理者に通知することができる。しかし、最近では攻撃者はまず IDS への攻撃を行って侵入を検出できないようにしてから本来の攻撃を行うことも増えてきた。そこで、IDS への外部からの攻撃を防ぐための手法として仮想マシンを用いた IDS のオフロードが提案されている。この手法は IDS を専用の仮想マシンで動作させ、そこから監視対象の仮想マシンをチェックする。これにより、IDS が攻撃される危険性を減らすことができる。

IDS のオフロードを行った際に生じる一つの問題が仮想マシンへのメモリ割り当てである。IDS をオフロードした場合には IDS はオフロード先の仮想マシンのメモリを使用する。そのため、オフロード元の仮想マシンはそれ自身に割り当てられたメモリに加えて、オフロードした IDS が消費するメモリも使えることになり、仮想マシン間での不公平が生じる。IDS が消費するメモリの分だけオフロード元の仮想マシンへのメモリ割り当てを減らすことも考えられるが、IDS が動作していない時には IDS 用のメモリを活用することができなくなる。

そこで本稿では、オフロードした IDS のメモリ使用量を考慮して仮想マシンへの動的なメモリ割り当てを行う Balloon Performer を提案する。Balloon Performer は IDS プロセスが使うメモリ量と IDS がファイルアクセスを行うことによって OS カーネル内に確保されるファイルキャッシュ量を定期的に測定する。取得したメモリ使用量に基づいて、オフロード元の仮想マシンへのメモリ割り当てを減らし、オフロード先の仮想マシンへと割り当てる。これにより、オフロードした IDS とオフロード元の仮想マシンが常に一定のメモリを使用するようにできる。我々は Balloon Performer を Xen 上に実装し、実験により仮想マシンへのメモリ再割り当てが正確に行えていることを確認した。

以下、2 章では、IDS のオフロードを行う際に生じるメモリ割り当ての問題についての説明する。3 章では Balloon Performer の設計とその実装について述べる。4 章では Balloon

†1 九州工業大学
Kyushu Institute of Technology
†2 独立行政法人科学技術振興機構, CREST
Japan Science and Technology Agency, CREST

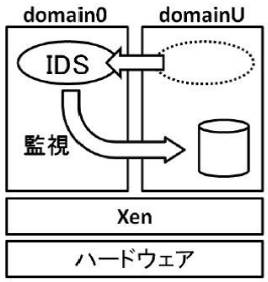

図 1 Xen における IDS のオフロード

Performer を用いたメモリの再割り当ての実験について述べる。5 章で関連研究に触れ、6 章で本稿をまとめる。

2. IDS オフロード時のメモリ割当の問題

Tripwire[1] はディスク内のファイルの整合性を定期的にチェックする IDS であるが、もし Tripwire が攻撃を受けた場合にはファイルの改竄を検出できなくなる恐れがある。Tripwire プロセスを停止されられると、ファイルの整合性のチェックが全く行われなくなる。Tripwire のポリシーファイルが書き換えられると、攻撃者が改竄したファイルを Tripwire がチェックしないようにすることができる。攻撃を受ける前のファイルのハッシュ値を記録したデータベースを改竄されると、ファイルを改竄した後のハッシュ値を正しい値だと Tripwire に誤認識させることができる。さらには、Tripwire の出力したレポートを書き換えることで、Tripwire 自身はファイルの改ざんを検出していても、管理者には通知されないようにすることができる。

このような問題を解決するために、仮想マシンを用いた IDS のオフロードが提案されている。この手法は IDS を別の仮想マシンで動作させることで、IDS への攻撃を防ぐことができる。Xen を用いてオフロードを行う際には、図 1 のように通常の仮想マシンであるドメイン U がオフロード元となり、ドメイン 0 と呼ばれる特権を持った仮想マシンがオフロード先となる。Tripwire をオフロードする場合、ドメイン 0 で Tripwire プロセスを動かし、Tripwire のポリシーファイル、データベース、レポートはドメイン 0 のディスク上に置かれる。ドメイン U の仮想ディスクイメージはドメイン 0 上に置かれるため、このイメージファイルをマウントすることでドメイン 0 からドメイン U のディスクの監視ができる。これにより、攻撃者がドメイン U に侵入したとしても、ドメイン 0 上で動く Tripwire プロセスを停止させたり、ドメイン 0 上に置かれたポリシーファイル等を改ざんしたりすることはできなくなる。ドメイン 0 では余計なネットワークサービスを動かさないため、ドメイン 0 が攻撃を受ける可能性は低い。

しかし、IDS をドメイン 0 にオフロードするとドメイン 0 のメモリを使って動作するようになるため問題が生じる。第一に、仮想マシン間の公平性が失われる。従来、ドメイン U には固定サイズのメモリを割り当てており、それ以上のメモリを使うことはできなかった。元々ドメイン U で動いていた IDS をオフロードすることでドメイン 0 のメモリを使うようになると、ドメイン U と IDS の合計でドメイン U に割り当てられたサイズ以上のメモリを使えることになる。これは IDS のオフロードを行わない仮想マシンに対して不公平である。

第二に、オフロードした IDS がドメイン 0 のメモリを使い過ぎるとシステム全体の性能が低下する恐れがある。Xen では、ドメイン U はドメイン 0 を介して I/O を行っている。そのため、ドメイン 0 のメモリが不足すると I/O 性能に影響が出る。この問題は特に、ドメイン U の数が増え、オフロードされる IDS の数も増えた時に生じやすい。特権を持ったドメイン 0 しかドメイン U の監視を行うことができないため、ドメイン U の IDS は必ずドメイン 0 にオフロードされる。

IDS が使用するメモリには、IDS プロセス自身が使用するメモリだけでなく、IDS がファイルを読み書きすることによって確保されるファイルキャッシュもある。ファイルキャッシュはプロセスがディスク内のファイルを読み込んだ時に OS カーネル内に作られ、以降のファイルアクセスを高速化するためにファイルの内容をメモリに保持しておく。 Tripwire を例に挙げると、Tripwire を実行した際に使用したメモリ量はスキャンしたファイル数 27,752 個に対して 528MB であった。そのうち、プロセスが使用したメモリ量は 121MB、ファイルキャッシュとして確保したメモリ量は 407MB となっており、Tripwire ではファイルキャッシュの量がプロセスの使用したメモリ量よりもかなり多いことが分かる。このことから、IDS が使用するメモリとして、プロセスの使用するメモリだけでなくファイルキャッシュも考慮することが重要であることが分かる。

このような問題を解決するために IDS が使用するメモリをあらかじめドメイン U から減らしておき、ドメイン 0 に割り当てるメモリを増やしておくという方法が考えられる。しかし、Tripwire のように IDS は定期的に実行されることが多いにも関わらず、ドメイン 0 で IDS が動作していない時でもドメイン U が IDS 用にドメイン 0 に多く割り当てたメモリを利用することができない。その結果、ドメイン U はオフロード前と比べるとより少な

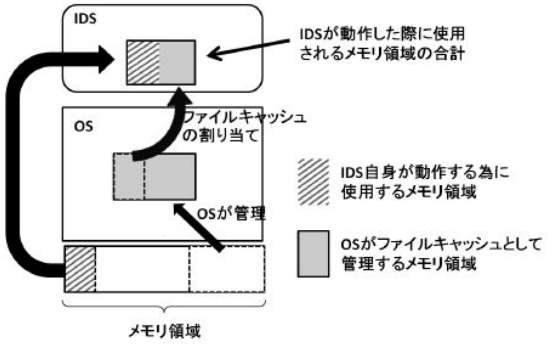

図 2 IDS の使用するメモリ

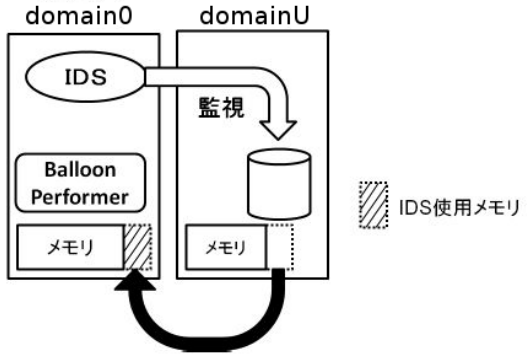

図 3 Balloon Performer 動作イメージ

いメモリしか使えなくなる。また、IDS が使用するメモリ量は一定とは限らないため、IDS が使用する可能性のあるメモリの最大量をドメイン U からドメイン 0 に移動させなければならなくなる。IDS があまりメモリを使っていない時でも、ドメイン U からはそのメモリを利用することができない。

3. Balloon Performer

3.1 概　　要

本稿では、オフロードした IDS が使用するメモリを考慮して仮想マシンのメモリを動的に割り当て直す Balloon Performer を提案する。Balloon Performer はオフロード先のドメイン 0 で実行されている IDS のメモリ使用量を定期的に測定する。そして、その分のメモリをオフロード元のドメイン U のメモリ割り当てから減らし、ドメイン 0 のメモリ割り当てを増やす。Balloon Performer では IDS プロセス自身が使用するメモリと IDS のファイルアクセスによって確保されるファイルキャッシュの両方を測定する。これにより、IDS がドメイン 0 で使用するすべてのメモリを考慮したメモリ割り当てを行うことができる。Balloon Performer は IDS のプロセス名とオフロード元のドメイン U を指定して実行され、定期的に IDS が実行されているかどうかの確認を行う。IDS が実行されていたらそのメモリ使用量に応じて仮想マシンへのメモリの再割当てを行う。

Balloon Performer を使用することにより IDS のメモリ使用量とドメイン U へのメモリ割り当ての合計を一定に保つことができるようになる。これにより、IDS のオフロードの有無に関係なく、仮想マシン間でのメモリ割り当てを公平に保つことができる。また、IDS をオフロードさせることによるドメイン 0 のメモリの圧迫を防ぐことができ、システム全体の性能が低下する可能性を減らすことができる。さらに、ドメイン 0 で IDS が動作していない時には、その分のメモリはドメイン U に割り当て直されるため、メモリをドメイン U が有効に活用することができる。

3.2 IDS のメモリ使用量の測定

ドメイン 0 にオフロードされた IDS が使用するメモリ量は Balloon Performer が定期的に測定を行う。

3.2.1 プロセスのメモリ消費量

プロセスの使用しているメモリ量は OS により記録されているため、OS から指定したプロセスのメモリ消費量を取得する。Xen のドメイン 0 の OS は Linux であるため、proc ファイルシステムからプロセスの情報を取得することができる。まず、pidof コマンドを用いて、オフロードされた IDS のプロセス名からプロセス ID を取得する。取得したプロセス ID を PID とすると、/proc/PID/status ファイルにプロセスのメモリ利用状況の詳細が記録されている。その中の VmRSS という項目から実際に使用されている物理メモリのサイズを取得する。IDS プロセスが動作していない時は pidof コマンドの結果が空になるため、プロセスのメモリ消費量を 0 とする。

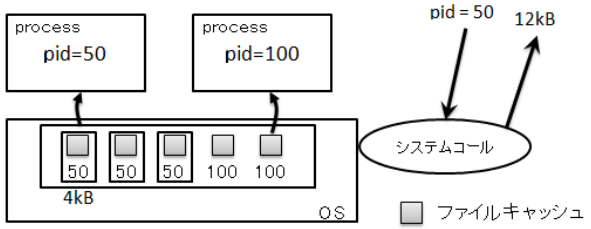

図 4 ファイルキャッシュとプロセスの対応づけ

3.2.2 ファイルキャッシュの使用量

プロセスが使用しているファイルキャッシュの量を取得できるようにするためには、ファイルキャッシュをプロセスに対応づける必要がある。しかし、従来の Linux にはどのプロセスがどのファイルキャッシュを使っているかという情報は記録されていない。一つの理由は、複数のプロセスが同じファイルにアクセスした場合、ファイルキャッシュがプロセス間で共有されることである。この場合、ファイルキャッシュをどのプロセスが使っているかの判断が難しい。もう一つの理由は、プロセス自体が終了してもファイルキャッシュはメモリ上に残されることである。このようなファイルキャッシュをどのように扱うかは難しい問題である。

そこで、Balloon Performer ではファイルキャッシュを確保したプロセスがその分のメモリを使用したものと考えて、プロセスとファイルキャッシュを一対一に対応づける。IDS が監視対象とするファイルはオフロード元のドメイン U のディスク上に置かれている。特定の IDS プロセスは 1 つのドメイン U のみを監視するため、同一のファイルを複数の IDS プロセスがアクセスする可能性は低い。そのため、多くのファイルキャッシュは特定のプロセスにだけ対応させることができる。また、IDS プロセスが終了した時には、ドメイン U の仮想ディスクをアンマウントすることで、その仮想ディスク上のファイルに関連したファイルキャッシュをすべて削除することができる。その結果、終了した IDS プロセスが使っていたファイルキャッシュがメモリ上に残らないようにすることができる。

我々はドメイン 0 の Linux カーネルがページキャッシュを確保する際に、カレントのプロセスの ID を記録するように変更を行った。Linux カーネルではファイルキャッシュはページキャッシュと呼ばれ、ページ単位で管理されている。メモリページの情報は page 構造体で管理されているため、そこにページキャッシュを確保したプロセスの ID を格納する pid というメンバ変数を追加した。そして、確保したメモリページをファイルキャッシュとして登録する add_to_page_cache 関数で page 構造体にプロセス ID を記録する。また、ファイルキャッシュをメモリから削除する際に呼び出される__remove_from_page_cache 関数で page 構造体の pid を初期化するように変更を加えた。

Balloon Performer が IDS プロセスの使っているファイルキャッシュ量を取得できるようにするために、プロセス ID を指定するとそのプロセスが使用しているファイルキャッシュの量を返すシステムコールも追加した。このシステムコールではすべてのメモリページに対応する page 構造体の pid メンバ変数をチェックし、対象のプロセス ID が記録されているページ数を返り値として返す。

3.3 仮想マシンへの動的メモリ割り当て

Balloon Performer は測定した IDS のメモリ使用量に基づいて、ドメイン U に元々割り当てられていたメモリのサイズから IDS のメモリ使用量を引いた値をドメイン U の新しいメモリサイズとして割り当て直す。ドメイン 0 に割り当てるメモリサイズは指定していないため、ドメイン U へのメモリ割り当てを減らすと、その分のメモリはドメイン 0 に自動的に追加される。ただし、ドメイン U に割り当てるメモリが少なくなり過ぎるとドメイン U が停止してしまうため、ドメイン U へのメモリ割り当てがしきい値を下回らないようにする。そのため、IDS がメモリを使いすぎる状況では、IDS のメモリ使用量とドメイン U に割り当てられたメモリサイズの合計が元々割り当てられたメモリサイズを超えてしまうことになる。この問題を解決するには、IDS が使うメモリ量を制限できるようにする必要があるが、それは今後の課題である。

ドメイン U に割り当てるメモリサイズを変更するために Xen Management API[2)]（以下、Xen API）を用いている。Xen API は Xen の管理を xm コマンドからではなくプログラムから行うために提供されている API である。Xen API を呼び出すと XML-RPC を使って xend との通信を行う。xend はハイパーコールを発行して仮想マシンへのメモリ割り当てを変更する。さらに、ドメイン 0 およびドメイン U のゲスト OS が使えるメモリ量を変更するために、ゲスト OS に組み込まれた balloon ドライバにメモリ量の調整を行わせる。ゲスト OS へのメモリ割り当てを減らす必要がある時には、balloon ドライバがメモリを確保して VMM にそのメモリを返す。一方、ゲスト OS へのメモリ割り当てを増やす必

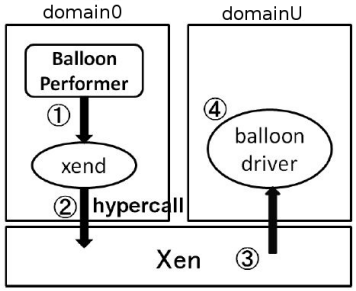

図 5 動的メモリ割当の流れ

要がある時には、balloon ドライバが確保したメモリを解放することで VMM にメモリを割り当てさせる。

Xen API を使ってドメイン U へのメモリ割り当てを変更するには、まず xend との間のセッションを確立するために xen_session_login_with_password 関数を呼び出す。次に、xen_vm_get_by_name_label 関数を用いてドメイン U の名前から VM オブジェクトを取得する。その後で、xen_vm_set_memory_dynamic_max_live 関数と xen_vm_set_memory_dynamic_min_live 関数を用いて、ドメイン U に割り当てるメモリ量を変更する。

4. 実　　験

実験には Intel Core2 Quad 2.83GHz の CPU および 4GB のメモリを備えた PC を用い、Xen 3.4.0 を動作させた。ドメイン 0 には 1.5GB のメモリを割り当て、ファイルキャッシュをプロセスに対応づけるように修正した Linux 2.6.18 を動作させた。ドメイン U にはメモリを 1.5GB 割り当て、準仮想化 Linux 2.6.18 を動作させた。

4.1 プロセスのメモリ消費量に応じた動的割り当て

Balloon Performer によるメモリの動的割当がプロセスのメモリ消費量を反映できていることを確認するために、5 秒おきにランダムにメモリの確保・解放を行うプログラムを動かし、プロセスのメモリ消費量とドメイン 0 に追加されるメモリ量を測定した。実験結果を図 6 に示す。この結果より、プロセスの消費メモリ量とドメイン U から減らされたメモリ量はほぼ一致しており、それらの合計がほぼ一定の 1.5GB になっていることが分かる。このことから、プロセスのメモリ消費に関しては Balloon Performer の動的なメモリ割り当てが正しく機能しているといえる。

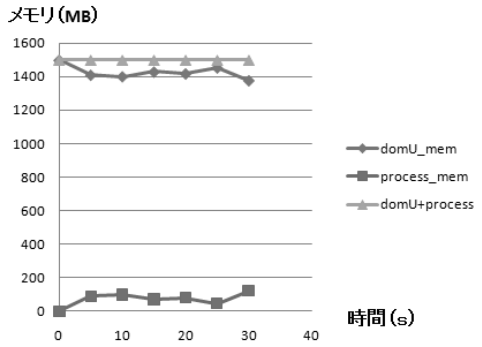

図 6 メモリ使用量に応じた動的割当

4.2 ファイルキャッシュに応じた動的割り当て

Balloon Performer によるメモリの再割り当てが IDS によるファイルキャッシュ使用量を反映できているかを確認するために Tripwire を用いて実験を行った。この実験では、Tripwire に 27,752 個のファイルをスキャンさせ、定期的にファイルキャッシュ使用量とドメイン 0 に追加されたメモリ量を調べた。実験結果を図 7 に示す。この結果より、プロセスのファイルキャッシュ使用量とドメイン U から減らされたメモリ量はほぼ一致しており、それらの合計はほぼ一定の 1.5GB になっていることがわかる。このことから、Balloon Performer によるファイルキャッシュを考慮したメモリの動的割当は正しく行われているといえる。

5. 関 連 研 究

OffloadCage[3] は仮想マシンを用いたセキュリティ機構のオフロードを考慮してスケジューリングを行う。セキュリティ機構をドメイン 0 にオフロードして動作させると、セキュリティ機構がドメイン 0 の CPU 時間を消費する。仮想マシンに一定の CPU 時間を割り当てて性能を分離している場合、元々ドメイン U で動いていたセキュリティ機構をドメイン 0 で動かすことで、合計すると割り当てより多くの CPU 時間を消費できることになってしまう。この問題を解決するために OffloadCage はオフロードしたセキュリティ機構とオフロード元の仮想マシンの CPU 時間の合計が一定になるようにスケジューリングを行う。Balloon

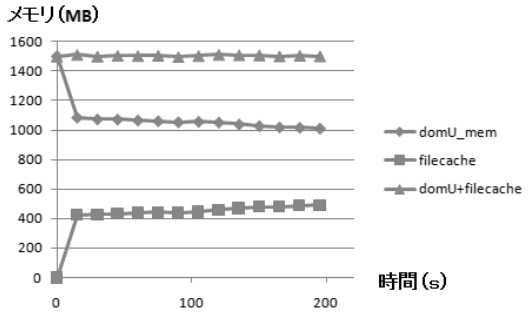

図 7 ファイルキャッシュを考慮した動的割当

Performer は CPU ではなくメモリについてセキュリティ機構のオフロード時の問題を解決している。

SEDF-DC[4] は Xen のスプリット・デバイスドライバを考慮して、仮想マシンの性能分離を実現する。Xen ではドメイン U が通信を行う時、ドメイン U 内のフロントエンド・ドライバがドメイン 0 内のバックエンド・ドライバと通信することでパケットの送受信を行う。このようにドメイン U の通信のたびにドメイン 0 のデバイスドライバが使用されるため、通信を多く行うドメイン U はドメイン 0 の CPU 時間を多く使えることになる。SEDF-DC ではパケットの送受信回数から消費される CPU 時間を推定し、その分をドメイン U への CPU 割り当てから減らす。ドメイン 0 のデバイスドライバが多くのメモリを使用する場合には、Balloon Performer でも考慮する必要がある。

Linux 2.6.23 で追加された cgroups の機能を用いると、cgroup 単位でページキャッシュを含めたメモリ使用量を取得することができる。IDS プロセスをある cgroup に所属させることで、IDS が使っているメモリ量が分かる。さらに、cgroup に割り当てるメモリ量を制限することもできる。この機能を利用することで、オフロードされた IDS のメモリ使用量を管理することができる。ただし、我々が実装に用いた Xen 3.4 のドメイン 0 の OS は Linux 2.6.18 であるため cgroups の機能は使えなかった。Xen 4.0 ではドメイン 0 の OS に Linux 2.6.32 が使えるため、Xen 4.0 を使うことで cgroups の機能を利用した実装を行うことができる。

6. ま と め

本稿ではオフロードした IDS のメモリ使用量に焦点を当てて、オフロードを考慮した仮想マシンへの動的なメモリ割当を行う Balloon Performer を提案した。Balloon Performer はオフロードした IDS プロセスのメモリ使用量と IDS のファイルアクセスによって OS カーネル内に確保されたファイルキャッシュ量の測定を行い、IDS が使用しているメモリ量をオフロード元のドメイン U からドメイン 0 に割り当て直す。これにより、IDS のメモリ使用量とドメイン U へのメモリ割り当ての合計を一定に保つことができる。Balloon Performer を用いることにより、IDS のオフロードを行ったとしてもドメイン 0 のメモリを圧迫せずに済み、IDS が使っていないメモリはドメイン U が活用することができる。実験により、IDS が使用するメモリをドメイン 0 に正確に割り当て直せていることが確認できた。

IDS のオフロード時にシステム全体の性能を考慮して仮想マシンにメモリを動的に割り当てられるようにすることが今後の課題である。現在の実装では、ドメイン U のメモリは元々の割り当ての半分以下にならないように制限しているが、ドメイン U の性能や正常な動作を測定しながらメモリ割り当ての最小値を決定する必要がある。また、IDS がドメイン 0 のメモリを使いすぎるのを防ぐために、IDS のメモリ使用量を制限する必要がある。特に、ファイルキャッシュの使用量を制御することが重要である。例えば、Tripwire の場合は多くのファイルに 1 回しかアクセスしないため、ファイルキャッシュは小さくても性能には影響しないといった特徴がある。そのため、特定のプロセスが使用するファイルキャッシュ量を制限できることが望ましい。

参 考 文 献

1) Tripwire:http://www.tripwire.com/
2) Xen Manegement API:http://wiki.xensource.com/xenwiki/XenApi
3) 新井昇鎬: セキュリティ機構のオフロードを考慮した仮想マシンのスケジューリング (2009).
4) Gupta, D., Cherkasova, L., Gardner, R. and Vahdat, A.: Enforcing Performance Isolation Across Virtual Machines in Xen (2006).