平成 21 年度　卒業論文概要			
所　　属	機械情報工学科	指導教員	光来　健一
学生番号	06237018	学生氏名	岡﨑 正剛
論文題目	セキュリティ機構のオフロードを考慮した 仮想マシンへの動的メモリ割当		

１．はじめに

インターネットに接続される計算機が増えるにつれて、不正アクセスやウィルスも増え続けてきた。それらに対抗する手段の一つとして侵入検知システム（IDS）がよく用いられている。IDS はネットワークパケットを解析したりファイルを監視したりすることで、攻撃者の侵入を検知する。IDS 自身への攻撃を防ぐために、近年、仮想マシンを用いて IDS をオフロードするという手法が提案されている。この手法は、IDS と監視対象のシステムを別々の仮想マシンで動作させることにより、IDS が攻撃を受けるのを防ぐ。しかし、IDS をオフロードするとオフロード先の仮想マシンのリソースを消費することになるため、これまではリソース割当をうまく管理することができていなかった。

本研究ではメモリリソースに焦点を当て、IDS のメモリ使用量を考慮して仮想マシンのメモリを動的に割当て直すシステム Balloon Performer を提案する。Balloon Performer はファイルキャッシュまで考慮して IDS のメモリ使用量を測定し、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にその分のメモリを移す。実験の結果、IDS が使用するメモリをほぼ完全に仮想マシン間で割当て直すことができた。

２．仮想マシンを用いたオフロードの問題点

IDS のオフロードは仮想化ソフトウェアを用いて実現される。仮想化ソフトウェアとして Xen を用いる場合、IDS は特権を持った仮想マシンであるドメイン 0 で動作し、それ以外の部分は通常の仮想マシンであるドメイン U で動作する。ドメイン 0 は全てのドメインの管理を行う権限を持っているため、ドメイン 0 で動作する IDS はドメイン U を監視することができる。

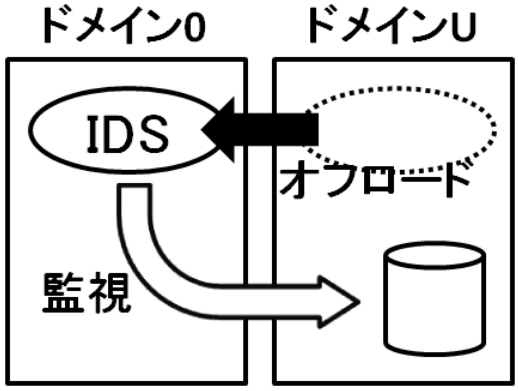

図１ IDS のオフロード

しかし、IDS をドメイン 0 にオフロードすることで、ドメイン 0 の CPU やメモリなどのリソースを消費してしまうという問題がある。様々な種類の IDS があるが、IDS によっては非常に重い処理を行ったり、大量のメモリを確保したりするものもある。また、直接メモリを確保しなくても、ファイルを読み込むとその内容を OS 内に保持しておくためのファイルキャッシュが作られる。大量にファイルを読み込んで監視を行う場合、ファイルキャッシュもドメイン 0 のメモリを圧迫する原因となる。ドメイン 0 は仮想マシンの実行において重要な役割を果たしているため、リソースが不足するとシステム全体の性能が低下する。

実際に IDS の一つである Tripwire がどの程度メモリを消費するかの調査を行ってみた。Tripwire はディスク上の大量のファイルを読み込み、その整合性をチェックする IDS である。Tripwire がチェックしたファイルの総数は 27,752 個であった。この調査の結果、Tripwire 自体が使用したメモリ量は最大 121MB、Tripwire の実行により OS 内に確保されたファイルキャッシュの量は 407MB であった。Tripwire をドメイン 0 にオフロードすることにより、最大 528MB のメモリが必要となることになる。この結果より、監視するドメイン U の数が増えると、すぐにドメイン 0 のメモリが足りなくなることが分かる。

３．Balloon Performer

本研究では、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割当を動的に変更する Balloon Performer を提案する。Balloon Performer は IDS が使用しているメモリ量を測定し、そのメモリ使用量の分だけオフロード元のドメイン U のメモリ割当を減らし、オフロード先のドメイン 0 のメモリ割当を増やす。この動的メモリ割当により、IDS が必要とするメモリをドメイン U からドメイン 0 に移すことができる。Balloon Performer のシステム構成を図２に示す。

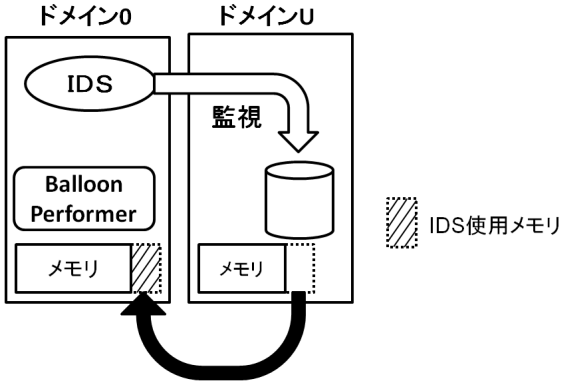

図２ Balloon Performer のシステム構成

３.１．プロセスのメモリ使用量の測定

Balloon Performer は IDS プロセスが使っているメモリ量

を proc ファイルシステムから取得する。Linux では proc ファイルシステム経由してプロセス情報を取得することができ、/proc/プロセス ID/status というファイルからプロセスの消費メモリ量を得ることができる。

一方、IDS の実行によって OS 内に確保された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、OS に変更を加えた。これは、プロセスが直接確保したメモリ量の統計情報は取られているが、どのくらいのファイルキャッシュを確保したかという情報は記録されていないためである。どのプロセスがどのファイルキャッシュを確保したかの情報を記録するために、個々のファイルキャッシュに確保したプロセスの情報を付加するようにした。この情報を基に、あるプロセスが確保したファイルキャッシュの量を返すシステムコールを作成した。

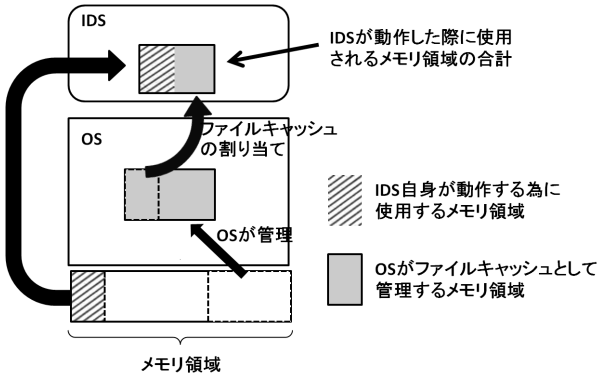

図３ IDS の使用メモリとファイルキャッシュ

３.２． 動的メモリ割当

Balloon Performer は XenAPI と呼ばれる機能を用いて、仮想マシンへのメモリ割当を動的に変更する。XenAPI を呼び出すとハイパーコールが発行され、Xen のメモリ割当機能が呼び出される。Xen は指定された仮想マシンの中の balloon driver にメモリの確保・解放を行わせる。 balloon driver は仮想マシン内の OS に組み込まれるデバイスドライバで、OS のメモリ管理アルゴリズムを用いて仮想マシンが使うメモリ量を増減させる。

４． 実験

Balloon Performer が IDS の使用しているメモリを考慮して動的にメモリ割当を変更できているかどうか調べるための実験を行った。この実験に用いたマシンは、CPU が Intel Core 2 Quad 2.83GHz、メモリが 4GB であった。初期状態として、ドメイン 0 に 1.5GB のメモリを割当て、ドメイン U に 1.5G B のメモリを割り当てた。

４.１． プロセスの消費メモリを考慮したメモリ割当

プロセスが使用するメモリ量を変化させて、ドメイン 0 へのメモリ割当がどのように変化するかを調べた。この実験を行うために、メモリ領域を確保・解放する malloc 関数と free 関数を用いて 15 秒ごとにランダムにメモリ領域の確保・解放を行い、自分自身のメモリ使用量を変動させるプログラムを作成した。

実験結果を図４に示す。この図は実験用プログラムが使ったメモリ量と、それに基づいて Balloon Performer がドメイン 0 の初期メモリ割当（1.5GB）に追加したメモリ量を 15 秒ごとにプロットしたものである。この２本のグラフは完全に重なっており、プロセスの消費メモリ量とドメイン 0 に追加されたメモリ量が一致していることを示している。このことから、Balloon Performer の動的メモリ割当はプロセスの使用したメモリ量を正しく考慮できていることが分かる。

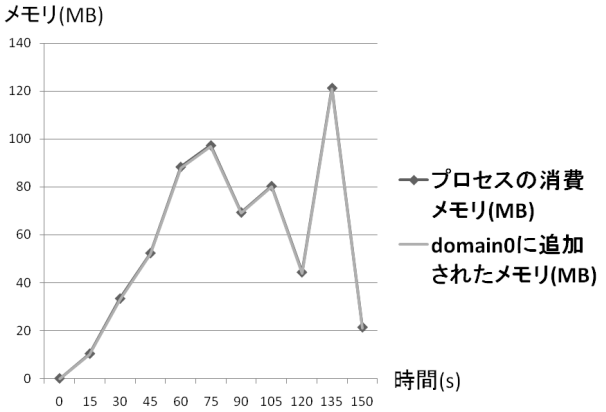

図４ 動的メモリ割当動作実験結果

４.２． ファイルキャッシュを考慮したメモリ割当

Tripwire の実行中に OS 内に確保されたファイルキャッシュの量に応じて、ドメイン 0 へのメモリ割当がどのように変化するかを調べた。この実験結果を図５に示す。この図は Tripwire のために確保されたファイルキャッシュの量と、それに基づいて Balloon Performer がドメイン 0 に初期メモリ割当に追加したメモリ量をプロットしたものである。この 2 本のグラフもほぼ完全に重なっており、プロセスの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がほぼ一致していることを示している。このことから、Balloon Performer の動的メモリ割当はプロセスの使用したファイルキャッシュも正しく考慮できていることが分かる。

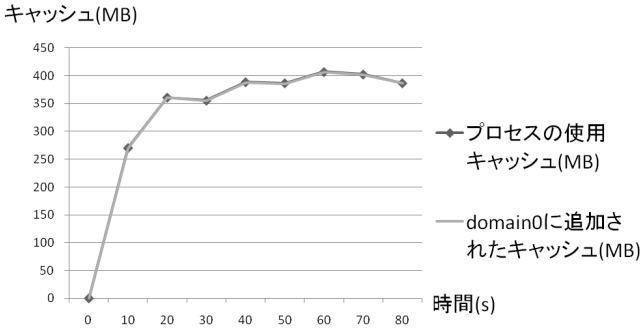

図５ 動的キャッシュ割当動作実験結果

５． まとめ

IDS のオフロードを考慮して仮想マシンに動的にメモリを割り当てるシステム Balloon Performer を提案した。Balloon Performer は IDS が使うメモリ量を測定し、オフロード元とオフロード先の仮想マシンのメモリ割当を調整する。実験の結果、本システムを用いることで IDS が使ったメモリを正確に考慮できていることが示された。今後の課題として、プロセスがドメイン U のメモリリソースを超えるかドメイン U が停止するほどのメモリを使用する場合に、プロセスの実行時間やドメイン 0 の未使用のメモリ領域などを考慮して、どのようなメモリ割り当てを行っていくのかを考える必要がある。