仮想マシンに対する高いサービス可用性を実現するパケットフィルタリング

安積 武志　田所 秀和　光来 健一　千葉 滋

本稿では仮想マシンモニタがゲスト OS 内の情報を用いてきめ細かいパケットフィルタリングを行うシステム xFilter を提案する。データセンタで仮想マシンを用いたホスティングが行われるようになるにつれて、スキルの低いユーザが仮想マシン内の OS を管理するケースが増えてきている。その結果、仮想マシンが踏み台攻撃に利用されたとしてもデータセンタの管理者が通信を遮断すべきだが、ゲスト OS の情報を利用できないために大雑把な通信制限しか行えなかった。xFilter はゲスト OS 内のどのプロセスがパケットを送信したかを仮想マシンモニタから解析し、攻撃を行っているユーザの通信だけを制限する。

1　はじめに

データセンタが仮想マシンを用いた仮想化ホスティングサービスを提供するようになったことで、ユーザは OS を自由に選択できるようになった。その一方で、十分にスキルの高くないユーザが OS を管理するケースが増えてきている。OS が正しく管理されていないと攻撃者の侵入を許してしまい、仮想マシンが踏み台攻撃に利用される可能性がある。その場合にはデータセンタの管理者が通信を遮断すべきである。しかし、従来はデータセンタ側からは仮想マシン内の OS の情報を利用することができなかったため、仮想マシン単位で通信を禁止するなどの大雑把な通信制限しかできなかった。これは仮想マシンに対してサービス可用性を低下させる原因となる。

本稿では、仮想マシンモニタからゲスト OS のパケットをプロセス単位、ユーザ単位でフィルタリングすることを可能にする xFilter を提案する。xFilter は仮想マシンモニタからゲスト OS 内のプロセス情報を取得し、プロセスの ID や所有者の情報を用いてパケットフィルタリングを行う。仮想マシンモニタからゲスト OS のプロセス情報を取得するために、仮想マシンモニタのメモリにゲスト OS のメモリをマッピングする。ゲスト OS の情報を用いることで、指定したプロセスやユーザの通信のみを遮断することができ、その他のプロセスやユーザは通信を行うことができる。また、xFilter はゲスト OS のユーザプロセスと行っている通信を一覧表示する機能を提供しており、異常を検出したときに攻撃を行っているプロセスやユーザを特定するのを支援する。我々は Xen 上に xFilter を実装した。

以下、2 章では既存の仮想マシンの通信制御の問題点について述べる。3 章では提案するフィルタリングシステムの詳細と実装を述べる。4 章では実験の結果を示す。5 章では関連研究について触れ、6 章で本稿をまとめる。

2　セキュリティとサービス可用性

データセンタがホスティングサービスとして仮想マシンを提供するようになったため、十分にスキルの高くないユーザが OS を管理するケースが増えてきている。従来のホスティングサービスではデータセンタが OS を用意し、ユーザはその上で動かすサービスだけ

Azumi Takeshi, Tadokoro Hidekazu, Kourai Kenichi, Chiba Shigeru, 東京工業大学, Tokyo Institute of Technology

を管理すればよい場合が多かった。仮想マシンへの攻撃を防ぐには、OS に最新のセキュリティパッチを適用し続け、ファイアウォールのルールなどを正しく設定する必要がある。しかし、OS を管理するユーザのスキルが低い場合には、システムの脆弱性を利用して攻撃者の侵入を許してしまう危険性がある。さらには、そこから外部のホストに対して攻撃を行うという踏み台攻撃の可能性もある。

そこで、我々はデータセンタの管理者が仮想マシンへの攻撃に対処できるようにすべきであると考える。仮想マシンのユーザは 24 時間 365 日管理しているとは限らないため、攻撃が検出された時に即座に対処できるとは限らない。また、即座に対処し始めたとしても、ユーザの管理スキルが低いと問題の解決に時間がかかってしまう。さらに、踏み台攻撃を行われると、仮想マシンのユーザだけではなく、データセンタの責任が問われることも考えられる。

データセンタ側での対処としては通信を制限することが考えられるが、仮想マシン環境では仮想マシンモニタと呼ばれるシステムソフトウェアでパケットフィルタリングするのが望ましい。従来はデータセンタと外部ネットワークとの間のファイアウォールで通信制限を行うことが一般的であった。この方法ではデータセンタ内の他のホストへの踏み台攻撃を防ぐのは難しい。仮想マシン内のゲスト OS が送信するパケットは必ず仮想マシンモニタを通過するため、仮想マシンモニタで通信制限を行うことで他の仮想マシンへの踏み台攻撃を防ぐことができる。

しかし、仮想マシンモニタからはゲスト OS 内の情報を利用できないため、通信の制御は大雑把になってしまい、仮想マシンに対してサービス可用性を低下させてしまう。例えば、メールサーバを動かしている仮想マシンに侵入されて、攻撃者が他のホストの 25 番ポートに対してポートスキャンを行っているという状況を考える。データセンタの管理者が行える最も単純な対処は、問題のある仮想マシンから外部への通信をすべて遮断してしまうというものである。この対処法では仮想マシンのサービス可用性を保つことができない。よりサービス可用性を保つ対処法としては、外部ホストの 25 番ポートへの通信だけを遮断するという対処法が考えられる。しかし、この場合でもメールサーバは外部にメールを送ることができなくなる。

3 xFilter

3.1 xFilter

我々は、仮想マシンモニタからゲスト OS のパケットをプロセス単位、ユーザ単位でフィルタリングすることを可能にする xFilter を提案する。xFilter は、仮想マシンモニタからゲスト OS の内部情報を取得し、プロセスの ID や所有者の情報を用いてパケットフィルタリングを行う。これにより、仮想マシンに対するサービス可用性を上げることができる。前章のメールサーバからの踏み台攻撃の例を考える。異常を検出したらまず、ゲスト OS の情報を参照して攻撃を行っているユーザを特定する。次にそのユーザが所有しているプロセスの行っている通信を調べ、指定したユーザの通信のみ遮断する。これにより、他のユーザは通常通りのサービスを受け続けることができる。

xFilter のフィルタリングの手順について説明する。まず仮想マシンモニタがパケットを受信したとき、ゲスト OS 内のどのプロセスがそのパケットを送信したかを調べる。ゲスト OS のメモリを仮想マシンモニタのメモリにマッピングすることによって、仮想マシンモニタから直接参照する。取得する情報は各プロセスのユーザ ID、名前、行っている通信のポート番号と IP アドレスの組である。仮想マシンモニタはパケットの到着毎にプロセス構造体が持つ通信の情報 (ポートや IP アドレス) と比較する。送信元のプロセスの ID またはユーザの ID がルールにマッチすれば送信を拒否する。

ルールを設定するために xFilter は攻撃元のプロセスやユーザの特定を支援する機能を提供している。外部へのポートスキャンなどの踏み台攻撃が検出されたら、上のようにして取得した通信、プロセス、ユーザの一覧を表示する。表示された情報と検出された通信を比較し、攻撃を行っているプロセスやユーザを特定する。同じプロセスからの攻撃が続いていればプロセス ID、同じユーザからの攻撃であればユーザ ID を指定する。

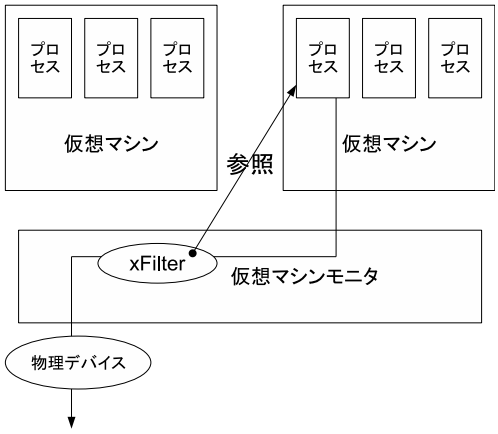

図 1　xFilter のアーキテクチャ

3.2　実装

xFilter の実装には、仮想マシンモニタとして Xen [1]、対象 OS として Linux を使用した。Xen において仮想マシンはドメインと呼ばれ、制御用の仮想マシンモニタとして動作するドメイン 0 とそれ以外のドメイン U が存在する。今回 xFilter は、ドメイン 0 上のユーザランドプロセスとして実装した。

ドメイン 0 からドメイン U のプロセスを調べるために、ドメイン U の仮想 CPU の GS レジスタからたどれるカレントプロセスから順に task_struct 構造体をたどっていく。Linux ではプロセスに関する情報は task_struct 構造体に格納されている。task_struct 構造体はリング状につながっているため、順番にたどっていけば全てのプロセスの情報を得られる。task_struct 構造体はオープンしているファイルを管理している file 構造体の配列へのポインタを持っている。さらに、file 構造体からファイルの実体を格納している inode 構造体の情報を得て、そのファイルがソケットであれば各ソケット固有の情報を持っている sock 構造体の情報を得る。sock 構造体にはそのプロセスの行っている通信のポート番号や IP アドレスなどの情報が格納されている。

ドメイン U のメモリにアクセスするために、ドメイン 0 からドメイン U のメモリを操作する機構 [5] を利用した。この機構を用いると、ドメイン U の仮想アドレスから Xen が管理するメモリフレーム番号を取得することができる。内部的には、ドメイン U のページテーブルを引いて物理アドレスを求め、そのアドレスを含むメモリフレームを取得している。さらに、ドメイン U 内の仮想的なメモリフレーム（疑似物理メモリ）を Xen が管理するメモリフレーム（マシンメモリ）に変換している。取得したメモリフレームをドメイン 0 のプロセスのアドレス空間に割り当てることで、ドメイン 0 のプロセスからドメイン U のメモリにアクセスすることができる。

現在のところ、xFilter によるパケットフィルタリングはドメイン 0 の iptables のルールを追加するという方法で実装している。Xen ではドメイン U の通信はすべてドメイン 0 を通過するため、ドメイン 0 のファイアウォールで制御が可能である。xFilter は一定時間毎にドメイン U のプロセスが行っている通信を調べる。その中から xFilter のルールで指定したプロセスやユーザが行っている通信を探し出し、その通信を拒否するように iptables のルールを自動的に追加する。追加した iptables のルールはログに保存することで、すでに追加されたルールを再度 iptables に追加しないようにしている。

4　実験

ベンチマークとして httperf [4] を用いて通信の性能を測定した。実験対象の計算機として、Athlon(tm) 64 Processor 3500+ の CPU、メモリ 1GB の計算機を使用した。仮想マシンモニタとしては、Xen3.1.0 、仮想マシンモニタ上で動く OS には Linux2.6.18 を用いた。ドメイン 0 にはメモリを 512MB、ドメイン U にはメモリを 256MB 割り当てた。

今回、フィルタリングにマッチしないルールを設定して、間隔を変えてポーリングを行った。条件は毎秒 150 リクエストを送り、総リクエスト数 100000 とした。実験は、xFilter を使わない場合、ポーリング間隔を 5 秒、3 秒、2 秒、1 秒にした場合の 5 通りで行った。実験結果は以下の表のとおりである。

ポーリングの間隔を短くすると、平均処理時間は長くなった。これは、ドメイン U のメモリを調べるときにドメイン U を停止しているためであると思われ

表 1　コネクションの処理時間 (ミリ秒)

polling 間隔	なし	5 秒	3 秒	2 秒	1 秒
最小	0.2	0.2	0.2	0.1	0.1
平均	0.4	0.4	0.5	0.5	0.7
最大	31.0	31.2	33.4	31.0	39.1
中央値	0.5	0.5	0.5	0.5	0.5
標準偏差	1.0	1.4	1.7	1.9	2.5

る。さらに、最小、最大、中央値はほぼ変わっていないにもかかわらず、ポーリング間隔が短くなるにつれて標準偏差が大きくなっていることを見ても、最大値に近い大きな値を取る回数が多くなっていることが分かる。つまり、ドメインの停止が処理にかかる時間に影響している。また実験所要時間は全て同じであり、つまりスループットには影響はなかった。これはサーバにまだ余裕があったためと思われる。

5　関連研究

Antfarm [2] は、仮想マシンモニタ上からドメインに手を加えずにプロセスの状態を取得する技術である。さらに、取得したプロセスの状態を使って仮想マシンモニタ上で anticipatory I/O scheduling を実装している。ドメイン上のオペレーティングシステムのソースコードに手を加えずに、オペレーティングシステムの情報を取得する点は本研究と同じである。しかし、取得できる情報はプロセスの状態の変化だけであり、何のプロセスかまでは分からない。オペレーティングシステムの内部構造まで解析していないので、取得できる情報は限られている。ドメイン上のオペレーティングシステムに依存せず、Linux 以外でも通用する技術であるという点で本研究より優れている。

Geiger [3] は、仮想マシンモニタ上からドメインに手を加えずにバッファキャッシュの状態を取得する技術である。Antfarm と同様に、ドメイン上のオペレーティングシステムのソースコードに手を加えずに、オペレーティングシステムの情報を取得する技術である。

6　まとめと今後の課題

本稿では仮想マシンモニタからゲスト OS 内の情報を取得し、利用できるシステムである xFilter を提案した。ゲスト OS 内の情報の取得は、ゲスト OS のメモリのマッピングによって実現した。また、パケットのフィルタリングには iptables を用いた。xFilter を用いることで、データセンタ管理者が OS 内の情報を利用してきめ細かい通信制御行うことができ、サービス可用性をできるだけ保つことができるようになった。

現在はポーリングを用いて実装しているが、ポーリングでは一度チェックしてから次のチェックまで一定の時間があり、その間の通信を制御することができない。そこで、ドメイン 0 のデバイスドライバを改造し、パケットが送られてきたときにドメイン U のメモリをチェックするように実装しているところである。

参考文献

[1] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A.: Xen and the art of virtualization, *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, New York, NY, USA, ACM Press, 2003, pp. 164–177.

[2] Jones, S. T., Arpaci-Dusseau, A. C., and Arpaci-Dusseau, R. H.: Antfarm: tracking processes in a virtual machine environment, *USENIX-ATC'06: Proceedings of the Annual Technical Conference on USENIX'06 Annual Technical Conference*, Berkeley, CA, USA, USENIX Association, 2006, pp. 1–1.

[3] Jones, S. T., Arpaci-Dusseau, A. C., and Arpaci-Dusseau, R. H.: Geiger: monitoring the buffer cache in a virtual machine environment, *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, New York, NY, USA, ACM, 2006, pp. 14–24.

[4] Mosberger, D. and Jin, T.: httperf-a tool for measuring web server performance, *SIGMETRICS Perform. Eval. Rev.*, Vol. 26,No. 3(1998), pp. 31–37.

[5] 田所秀和, 光来健一, 千葉滋: 仮想マシン間にまたがるプロセススケジューリング, **情報処理学会論文誌:コンピューティングシステム**, ACS 23, 掲載予定.