ゲストOS情報を用いたVMMレベル・パケットフィルタ

安積武志† 光来健一††,††† 千葉滋†

1. はじめに

クラウドコンピューティングなどのために利用者に仮想マシン（VM）を提供しているデータセンターにとって、踏み台攻撃は大きな脅威である。このタイプの攻撃は、攻撃者自身のホストからではなく、VM内のソフトウェアの脆弱性を利用し、VMに侵入して攻撃を行う。そのためデータセンターは、攻撃の被害者であると同時に加害者にもなってしまう。データセンターにとって、できるだけ早く踏み台攻撃を止めることが重要である。データセンターのファイアウォールでパケットフィルタリングを行えば、安全で効果的である。この方法では、侵入されたVMからの踏み台攻撃を完全に遮断することができる。しかし、踏み台となったVMのサービス可用性が著しく低下する。攻撃パケットを遮断するだけでなく、正常なアプリケーションまでパケットを送信できなくなってしまう。

そこで我々は、仮想マシンモニタ（VMM）内できめ細かいパケットフィルタリングを行うxFilterを提案する。xFilterはVMのメモリを解析してゲストOS内部の情報を取得し、できるだけ攻撃パケットだけを遮断する。例えば、パケットの送信元を特定することで、攻撃プロセスから送信されたパケットだけを遮断できる。

2. 踏み台攻撃への従来の対処法

外部ファイアウォールによるパケットフィルタリングは踏み台にされたVMの外部にあるため、侵入者が攻撃することは困難であり、安全であるといえる。しかし、IPアドレスやポートといったパケットに含まれる情報のみを元にフィルタリングを行うため、踏み台VMのサービス可用性が著しく低下する。例えば、メールのSPAM攻撃が検出された時、25番ポートへのパケットをすべて遮断すると、そのVM内のすべてのアプリケーションがメールを送信できなくなる。SPAM攻撃は防ぎつつ、正常なアプリケーションはメールを送信できるようにするのが望ましい。

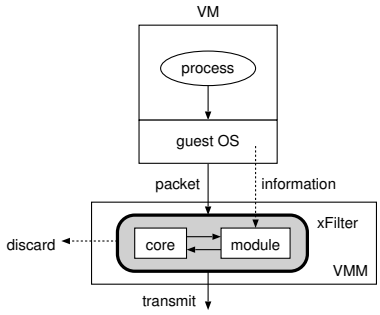


図1 VMM内で動作するxFilter

他方、踏み台VM内のファイアウォールを利用すると、高いサービス可用性を実現できる。このファイアウォールはOSカーネル内に存在するので、フィルタリングに送信元プロセスの情報を利用することができるためである。しかし内部ファイアウォールは侵入者の攻撃を受けやすく、攻撃パケットを遮断するルールを削除することも容易である。

3. xFilter

安全性とサービス可用性を両立した踏み台攻撃への対処法として、我々はxFilterを提案する。

3.1 VMM内でのきめ細かいフィルタリング

xFilterはVMM内で動作するパケットフィルターである。VMMはVMとは隔離されており、VMへの侵入者がVMM内にあるxFilterを攻撃することは困難である。さらに全てのパケットはVMMを通過するので、VMからのネットワークパケットを全て検査することができる。

xFilterはVMのメモリを解析することでゲストOS内の情報を利用し、サービス可用性を向上させている。Linuxの場合、task_struct構造体がプロセスを管理しており、ソケット情報も保持している。これらの構造体をたどって調べることで、特定のパケットを送信し

† 東京工業大学
Tokyo Institute of Technology
†† 九州工業大学
Kyushu Institute of Technology
††† 独立行政法人科学技術振興機構, CREST

たプロセスが分かる。この情報をフィルタリングルールに指定することにより、特定のプロセスやユーザが送信したパケットだけを遮断することが可能になる。

xFilter は開発者がフィルタモジュールを実装することで、ゲスト OS 内の様々な情報を利用したフィルタリングを実現することができる。xFilter は図??本体とモジュールで構成されている。VM 内のプロセスがパケットを送信したら、VMM がそれを横取りして xFilter 本体に渡す。xFilter 本体はモジュールを呼び出し、IP アドレスやポートなどのパケットのヘッダ情報を渡す。フィルタリングルールとメモリ解析によってゲスト OS から取得した情報に基づいて、パケットフィルタの結果を xFilter 本体に返す。その結果に基づいて、xFilter 本体はパケットを送信するか破棄する。

3.2 xFilter モジュールの開発サポート

xFilter モジュールは VMM 内に実装されるため、その開発は容易ではない。少しでも変更すれば、新しいモジュールを有効にするためにシステム全体を再起動する必要がある。またモジュールにバグがあった場合、VMM 全体がクラッシュしてしまい、システム全体を再起動することになってしまう。

そこで xFilter では、ヘルパー VM と呼ばれる別 VM 上のプロセスとしてモジュールを動作させることを可能としている。モジュールがクラッシュしてもプロセスを再起動するだけでよく、他のシステムにも影響を及ぼさない。対象 VM のメモリ解析は、VMM の提供している機能を利用する。新しいモジュールのデバッグが終わったら、変更なしに VMM に組み込むことができる。再コンパイルだけで済ませるために、xFilter は、VMM とヘルパー VM の両方のモジュールで同じ API を提供している。

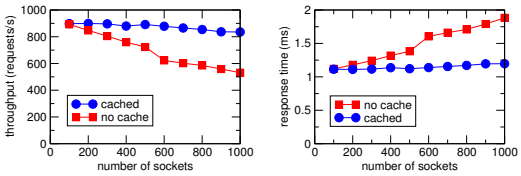
3.3 フィルタリングルールの自動最適化

送信パケットに対して NIDS を適用し、攻撃が検出されると自動的にフィルタリングルールを追加する。この際に、xFilter モジュールがメモリ解析によってそのパケットの送信元プロセスを特定し、そのプロセス ID を指定したルールを生成する。

また、xFilter は自動でフィルタリングルールの最適化を行う。送信元プロセスが次々と変わるような攻撃の場合、送信元プロセスをルールに追加し続けるとルールが肥大化してしまう。送信元プロセスが異なってもそれらのプロセスの所有者が同一であれば、xFilter はユーザ ID を指定するルールに置き換える。

4. 実　　験

我々は xFilter を Xen 3.4.2[1)] に実装し、xFilter の



(a) Throughput　　(b) Response time

図 2 xFilter によるウェブサーバの性能低下

オーバーヘッドを調べる実験を行った。サーバマシンの CPU は Intel Core i7 860、メモリは 8GB、NIC はギガビットイーサーネットであった。メモリは、ドメイン 0 に 7GB、ドメイン U に 1GB 割り当て、ゲスト OS は Linux 2.6.18 を動かした。

実際のアプリケーションにおける性能の低下を調べるため、ApacheBench ベンチマークを用いて Apache ウェブサーバのスループットとレスポンスを測定した。xFilter のオーバヘッドはソケットの数に依存するため、ソケットの数を様々に変えて性能を測定した。スループットとレスポンスは図 2 のとおりである。パフォーマンス低下はソケット数に比例していることが分かる。すべてのパケットについてゲスト OS のメモリ解析を行った場合（no cache）、1000 ソケットでの性能低下はスループットで 43%、レスポンスで 76%になった。ただし、同一コネクション内のパケットに対してはメモリ解析を行わないようにするキャッシュを有効にすると、スループットの低下は 10%、レスポンスの低下は 12%であった。100 ソケットでは、性能低下は 3.9%になった。

5. ま と め

本稿では、VMM 内で行うきめ細かいパケットフィルタリングシステム xFilter を提案した。踏み台 VM のサービス可用性を向上させるため、VM のメモリ解析を行うことで、ゲスト OS の情報をフィルタリングに利用できるようにした。実験では、xFilter のオーバーヘッドがソケット数に比例しているが、100 ソケット程度では 4%以下であることを示した。

参 考 文 献

1) P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A.Warfield. Xen and the Art of Virtualization. In *Proc. Symp. Operating Systems Principles*, pp. 164–177, 2003.