L-022

仮想マシンモニタにおけるデバイスドライバ安全性向上に関する提案

藤澤 一樹 † 橋本 正樹 † 宮本 久仁男 † 金 美羅 † 辻 秀典 ‡ 田中 英彦 †
Kazuki Fujisawa Masaki Hashimoto Kunio Miyamoto Mira Kim Hidenori Tsuji Hidehiko Tanaka

1. はじめに

近年，仮想化技術が注目を集めているのは，仮想化技術を用いて複数台存在したサーバを１台に集約することによりサーバの運用効率を向上させることで TCO（Total Cost of Ownership）の削減ができるという事による．仮想化技術の中でも特に仮想マシンモニタ（VMM：Virtual Machine Monitor または Hypervisor）と呼ばれる種類のソフトウェアが注目を集めており，その代表的な例として Xen や KVM（Kernel Based Virtual Machine）などの OSS（Open Source Software）が挙げられる．

VMM は，VMM 上で稼動する VM について使用するリソースを隔離できるとともに，Guest OS の実行環境を防御することができるため，これらの機能をセキュリティ分野に適用した研究例が多く存在する．しかしながら，これらの研究は VMM が安全であるという前提に基づいているため，VMM に脆弱性が存在するとその上に載せた Guest OS が安全であってもシステム全体は脆弱になってしまう．そこで本稿では，VMM の脆弱性を指摘し，安全性を向上させる手法を提案する．

2. VMM とセキュリティ

現在，VMM の脆弱性として考えられるのは，デバイスドライバの脆弱性を突いて VMM の特権を取得することである．実装によるが，VMM は OS と同様に特権モードでデバイスドライバを動作させるため，デバイスドライバに脆弱性があった場合，VMM における特権命令を実行することができる．これによって Guest OS の停止や新たな Guest OS を構築し踏み台にされるなどの問題が起きる．また，現在の計算機環境においては，多種多様のデバイスが存在し，各デバイスが多様の機能を持つため，これらのデバイスを制御するデバイスドライバはコードが大きくなり，脆弱性を持つことが知られている．このため，デバイスドライバが持つ脆弱性によって起きる問題を事前に防ぐ必要がある．

3. 関連研究

デバイスドライバの脆弱性によって起きる問題を回避する手法として，特権モードで動作するデバイスドライバをできる限り少なくする方法 [3][4][5] や，デバイスドライバそのものの脆弱性を無くす方法 [1][2] が挙げられる．

Microsoft 社は上記の両方の手法をとっている．まず，デバイスドライバ開発者に対してデバイスドライバの脆弱性検査ツール [1] を提供している．また，デバイスドライバを Microsoft 社が検証し，問題なければそのドライバに署名を行う [2]．さらにカーネルモードデバイスドライバとユーザモードデバイスドライバ [3] に分け，殆どの処理をユーザモードで行うようにしている．デバイスドライバの脆弱性検査ツールについては有効であるが，デバイスドライバ署名については OSS では誰が保証するのかが難しく OSS で実現するのは困難である．ユーザモードドライバに関しても Xen や KVM は Linux の構造を基にしているため大幅なソースコードの変更が必要となる．

Terra[4] ではデバイスドライバがセキュリティ上の問題を起こすことを指摘し，解決策としてユーザモードドライバと Microsoft の NGSCB(Next-Generation Secure Computing Base) を利用することを提案している．

Xen[5] では Xen を管理する特権ドメイン（dom 0）と非特権ドメイン（dom U）に分けて，dom U がデバイスにアクセスする場合，必ずフロントエンドドライバから Xenbus を通ってバックエンドドライバにアクセスし dom 0 のデバイスドライバがデバイスにアクセスする．Xen 自体はデバイスドライバを持たないが，Xen を管理する dom 0 がデバイスドライバを持っているために根本的な解決手法にはならない．

そこで本研究では VMM におけるデバイスドライバの安全性を向上させる手法を提案する．

4. 提案システム

デバイスドライバは OS がデバイスを制御するためのインタフェースである．OS とデバイス間のデータのやりとりは，独立した I/O メモリ空間，またはメモリマップド I/O が使われる [6]．独立した I/O アドレス空間を使う方法では，一連の I/O 命令と特殊な I/O 保護機構が使え，逐次実行ができる．メモリマップド I/O では，物理メモリ空間を使用し，セグメンテーションとページングの保護機構とプロセッサの汎用移動命令を使用する方法がある．しかしながら，デバイスドライバは特権で動作するため脆弱性を持つと大きな脅威となる．以下にデバイスドライバの脆弱性を挙げる．

- 特権ユーザでなければならない操作を非特権ユーザに許すこと
- デバイスドライバはカーネルのメモリ空間を利用するため初期化しないことによるデータのリーク
- バッファオーバラン攻撃

これらの脆弱性が突かれると以下の問題を起こす.

- カーネルメモリ空間のデータリーク
- DoS 攻撃
- 非特権ユーザの特権取得

本稿では，これらの問題を防止するためにデバイスドライバに対するリファレンスモニタの導入するシステム

†情報セキュリティ大学院大学
‡株式会社情報技研

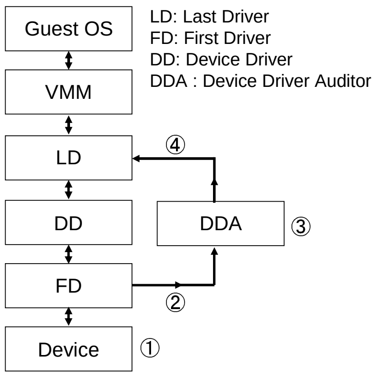

図 1: 提案システム

を提案する．提案システムは Flask[7] アーキテクチャを採用し，入出力に対して強制アクセス制御を行う．図 1 は提案システムの動作手順を現わしている．

FD（First Driver）と LD（Last Driver）は Object Manager として動作する．FD は，デバイスからのアクセスに対してデータや割り込みなどをチェックし，メッセージとして DDA に送る．また，VMM からのアクセスに対して DDA のポリシーに基づいて強制アクセス制御を実行する．LD は FD の動作と逆の動作を行う．VMM からのアクセスに対してその権限をチェックし，メッセージとして DDA に送る．DDA（Device Driver Auditor）は Security Server として FD と LD からのメッセージを受け取り，ポリシーに違反しているかどうかをチェックする．

Step1: デバイスから割り込み線に信号を受信するとハードウェア割り込みがかかる

Step2: デバイスの動作を検出するために FD がデバイスドライバへ処理が依頼されることを検出する

Step3: その情報を DDA に送る．DDA はポリシーに定義された問題のある動作を引き起こす場合に LD に対して防御するよう LD に指令を出す

Step4: DDA から通知されると LD がアクションを起こして防御する

提案システムでは，I/O アクセスの際に必ずリファレンスモニタを通過する．即ち，デバイスからデバイスドライバに対するアクセスは必ず FD を通ることになり，また逆に VMM からデバイスへのアクセスは必ず LD を通ることになる．これにより，上記の脆弱性を防ぐことができる．但し，リファレンスモニタは以下の要件を満たす必要がある．

- リファレンスモニタは全てのアクセス要求に対して必ず呼び出されること
- リファレンスモニタのメカニズムは破壊や改竄を受けないこと
- リファレンスモニタが正しく動作することが保証されていること

5. まとめと今後の課題

本稿では VMM における問題点を指摘し，その対策としてデバイスドライバに対するリファレンスモニタを提案した．本手法を利用することで，ネットワークパケットを大量に放出するような Warm や rootkit などが検出可能になると考えられる．ただし，ここで検出可能な rootkit はデバイスドライバを改変してしまうような rootkit に限定する．提案システムは最も基本的なものであり，今後の課題として以下の項目が挙げられる．

- リファレンスモニタの要件を満たす実装
- I/O 処理の増加によるパフォーマンスの低下への対策
- LD，FD，DDA の設置メモリ空間の検討
- 監査を行うタイミングの検討
- 監査後の動作の検討

参考文献

[1] Microsoft, “テストツール：はじめに”. http://www.microsoft.com/japan/whdc/devtools/tools/toolstart.mspx

[2] Microsoft, “ロゴ及び WHQL テスト”. http://www.microsoft.com/japan/whdc/GetStart/default.mspx

[3] Microsoft “Windows Driver Foundation (WDF)”. http://www.microsoft.com/japan/whdc/driver/wdf/default.mspx

[4] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, Dan Boneh, “Terra: A Virtual Machine-Based Platform for Trusted Computing”, ACM SIGOPS Operating Systems Review, Vol.37, No.5, pp.193-206, 2003.

[5] Keir Fraser, Steven Hand, Rolf Neugebauer, Ian Pratt, Andrew Warfield, Mark Williamson, “Safe Hardware Access with the Xen Virtual Machine Monitor”, OASIS ASPLOS 2004 workshop.

[6] Intel, “Intel ®64 and IA-32 Architectures Software Developer’s Manual Volume 1: Basic Architecture”, May, 2007.

[7] Ray Spencer, Peter Loscocco, Stephen Smalley, Mike Hibler, David Anderson, and Jay Lepreau, “The Flask Security Architecture: System Support for Diverse Security Policies”, Proceedings of the 8th conference on USENIX Security Symposium, pp.123-139, August 1999.