動作中のOSを安全な仮想マシン上に移行するための仮想マシンモニタ

野元 励[†] 大山恵弘[††]

1. はじめに

オペレーティングシステム（OS）のカーネルが持つバッファオーバーフロー脆弱性が問題となっている．カーネルがバッファオーバーフロー脆弱性を持つと，これを利用したバッファオーバーフロー攻撃により，攻撃者がカーネルの特権レベルで任意のコードを実行することが可能になる．その結果，OSの動作に悪影響を及ぼすだけでなく，情報漏洩の発生や，他者のコンピュータに対する不正アクセスへ利用される危険性がある．

一方で近年，OSのセキュリティ向上を目的とした仮想マシンモニタ (VMM) の研究が行われている．これは，ハードウェアとOSの間に，VMMを組み込むことでOSの挙動を制御し，OSのセキュリティを向上させるものである．しかし，セキュリティの向上のための既存のVMMは，VMMのインストールにOSを変更する必要や[2]，OSを起動させる前にVMMを起動する必要があり[3]，サービス提供中のサーバマシンなどには適用が難しいという問題がある．

そこで本研究では，動作中のOSに対して，再起動なしに組込み可能なセキュリティ向上のためのVMM，HyperShieldを提案する．HyperShieldは，CPUによる仮想化支援技術であるIntel VT-x[1]を利用した小さな仮想マシンモニタ（VMM）として実現した．このVMMはOSの上で実行され，実行されると動作中のOSを仮想マシン（VM）上へ移行する．そして，VMのメモリ管理を通じて，CPUがカーネルの特権レベルである場合には，ユーザ空間上のコードを実行禁止にする．これにより，VM上で動作しているOS（ゲストOS）では，カーネルバッファに対するバッファオーバーフロー攻撃によるユーザコードの実行は無効化される．また，対象OSはLinuxとし，Loadable Kernel Module（LKM）として実現した．

† 電気通信大学大学院電気通信学研究科情報工学専攻
†† 電気通信大学情報工学科

2. Intel VT-xの概要

Intel VT-x[1]は，CPUによる仮想化を支援する機能であり，VMMの作成を容易にする．Intel VT-xの機能を利用すると，CPUはVMMが動作するためのVMX rootオペレーションモードと，ゲストOSが動作するためのVMX non-rootオペレーションモードの二つの動作モードが利用できる．

VMX rootオペレーションモードは通常のプロテクトモードとほぼ同様であるが，この動作モード専用の命令でVMX non-rootオペレーションモードへ移行することができる．この移行をVM entryと呼ぶ．VMX non-rootオペレーションモードでは，指定した外部割込みや例外，特定の命令の実行などにより，自動的にVMX rootオペレーションモードへ移行する．この移行をVM exitと呼ぶ．VMMはVM exitが起こるとその原因を調べ，必要なエミュレーション処理を行った後にVM entryを行う．これを繰り返すことで，ゲストOSを容易に動作させることができる．

3. 設計と実装

3.1 基本設計

HyperShieldは，Intel VT-xを利用し，メモリ管理とCPUのみを仮想化するVMMである．メモリアクセスはVMMが管理し，I/Oなどのデバイスへのアクセスはゲスト OSが直接行う．また，デバイスドライバを組み込む以外にゲストOSに変更を加えない．

3.2 実装の概要

HyperShieldは，VMX rootオペレーションモードのring0で動作しているOSをVMX non-rootオペレーションモードのring0で動作させることでゲストOS化する．HyperShield自体は，VMX rootオペレーションモードのring0で動作し，VMX non-rootオペレーションモードで動作しているOSの制御を行う．

VMMの挿入とゲストOS化は，LKMの初期化処理内で行う．初期化処理では，現在のCPU情報を取

得し，それを VM の CPU 情報として設定する．但し，VM の EIP の値は，VM entry を行う次の命令のアドレスとする．そして，VM entry を行うと，次の命令からゲスト OS として処理が再開される．

3.3 メモリの仮想化と防御機能の実装

VMM は OS が作成したページテーブルを参照しながら，独自のページテーブルを作成する．以下では，これをシャドウページテーブルと呼ぶ．シャドウページテーブルは，OS が作成したページテーブルと同じ物理アドレスを指し，OS が作成したページテーブルからのアドレス変換は行わない．

防御機能の実現には，ページテーブルに実行権限を設定するための機能である eXecute Disable ビット（XD ビット）を利用する．VMM は，ゲスト OS がカーネルの特権レベルで動作していると，シャドウページテーブルのユーザ空間の全メモリに XD ビットによる実行保護を設定する．この設定により，ゲスト OS がユーザモードの特権レベルに移行し，ユーザ空間のメモリにアクセスしようとすると，実行保護違反によるページフォルトが発生する．VMM がこのページフォルトを検知すると，シャドウページテーブルに設定した実行保護を解除する．但し，実行保護違反によるページフォルト発生時に，カーネルモードの特権レベルでユーザ空間のメモリを実行しようとした場合は，バッファオーバーフロー攻撃が起こったものと判断する．この場合は，ゲスト OS にバッファオーバーフロー攻撃の通知を行い，さらにゲスト OS にページフォルト例外を挿入する．これにより，バッファオーバーフロー攻撃を防御する．

4. 評　　価

4.1 防 御 実 験

HyperShield を用いて，バッファオーバーフロー攻撃についての防御実験を行った．この実験では，デバイスドライバの脆弱性を利用して，ユーザプログラム上で特権命令である CR4 に対する MOV 命令を実行する攻撃を行った．HyperShield 組込み前では，攻撃により CR4 の値が書き換わったことが確認できたが，HyperShield 組込み後では，攻撃による CR4 の値の書き換えは起きなかった．以上のことから，HyperShield によってカーネルに対するバッファオーバーフロー攻撃の防御が確認できた．

4.2 性 能 評 価

ベンチマークとしては，Unixbench 4.10 と Linux カーネルのコンパイルを用いた．Unixbench の測定結果を**表 1** に，カーネルコンパイルの実行時間を**表 2** に示す．

表 1 Unixbench のスコア

	execl	syscall	spawn
実マシン	4381.8	429730.7	9014.9
HyperShield	438.1	386553.4	149.8

表 2 カーネルコンパイルの実行時間（s）

	実行時間
実マシン	227.17
HyperShield	481.87

Unixbench では，`execl`，`syscall`，`spawn`，`context1` の 4 種類について測定した．`execl` は `execl` 呼出し，`syscall` はシステムコール呼出し，`spawn` はプロセス生成，`context1` はコンテキストスイッチについての測定である．表 1 より，最大で 65 倍程度のスコアの差がついた．これはプロセス生成やプロセス切替えに伴う，シャドウページテーブル作成が大きなオーバーヘッドになっているためだと考えられる．表 2 では，HyperShield の導入によってカーネルコンパイルの実行時間は約 2.1 倍に伸びている．これは，カーネルコンパイルでは I/O による時間が多くを占めるため，シャドウページテーブル作成のオーバーヘッドの相対的な大きさが，小さくなったためである．

5. 今後の課題

今後の課題としては，シャドウページテーブルの管理に伴うオーバーヘッドの低減や VMM 自体の保護機能が挙げられる．

参 考 文 献

1) Intel Corporation. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, September, 2008.
2) Arvind Seshadri, Mark Luk, Ning Qu, Adrian Perrig. SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes. In *Proceedings of 21st ACM Symposium on Operating Systems Principles*, pages 335-350, Washington, October, 2007.
3) 品川 高廣, 榮樂 英樹, 谷本 幸一, 面 和成, 長谷川 晶一, 保理江 高志, 平野 学, 光来 健一, 大山 恵弘, 河合 栄治, 河野 健二, 千葉 滋, 新城 靖, 加藤 和彦. 準パススルー型仮想マシンモニタ BitVisor の設計と実装, SWoPP 佐賀 2008, 佐賀, 8 月, 2008.