# 全体ミーティング(5/13)

# Implicit Detection of Hidden Processes with a Feather-Weight Hardware-Assisted Virtual Machine Monitor

Yan Wen, Jinjian Zhao, Huaimin Wang, Jiannong Cao

[ACISP'08]

# Introduction

- Rootkit等のmalwareへの対策
  - "Cross-view validation": 複数の視点からHidden Processの有無を確認する
  - ex. Kernel上のデータ構造と直接比較する手法
    - Privileged malwareに対応できない
  - ex. VMM-layerで検出する手法
    - OS-level informationをVMM-Layerで把握する必要がある (semantic gap)
  - 本論文では上記問題を解決するDynamic Migrationについて説明し、Hidden Processの検出が可能であることを示す

# Overview

- AriesVM
  - Dynamic Migration
    - Bare hardware上で動作するOSの下にVMMを動的に挿入する
  - Detect Hidden Process via TPL(True Process List)
    - Kernelから見た(操作済の)Process ListとHardware Levelで見たProcess Listを比較

# Dynamic Migration(1)

- 動作中のOSを動的にVMM上にMigrationする
  - HardwareとOSの間にVMMが動的に挿入されるイメージ
  - 取り外しもできる
- Intel VT等のHardware-assist VM機構を利用
  - 本実装ではIntel VTを利用
    - VMMが動作する "root-mode"
    - VMが動作する "non-root mode"
  - 上2mode間を切り替えることで省コストなVMの実装を行える
    - x86特有の面倒な仮想化処理はCPUにお任せ
- テクニック自体の初出はBlue Pill[J.Rutkowska, Blackhat'06](だと思っている)
  - 論文中ではReferenceなし

# Dynamic Migration(2)

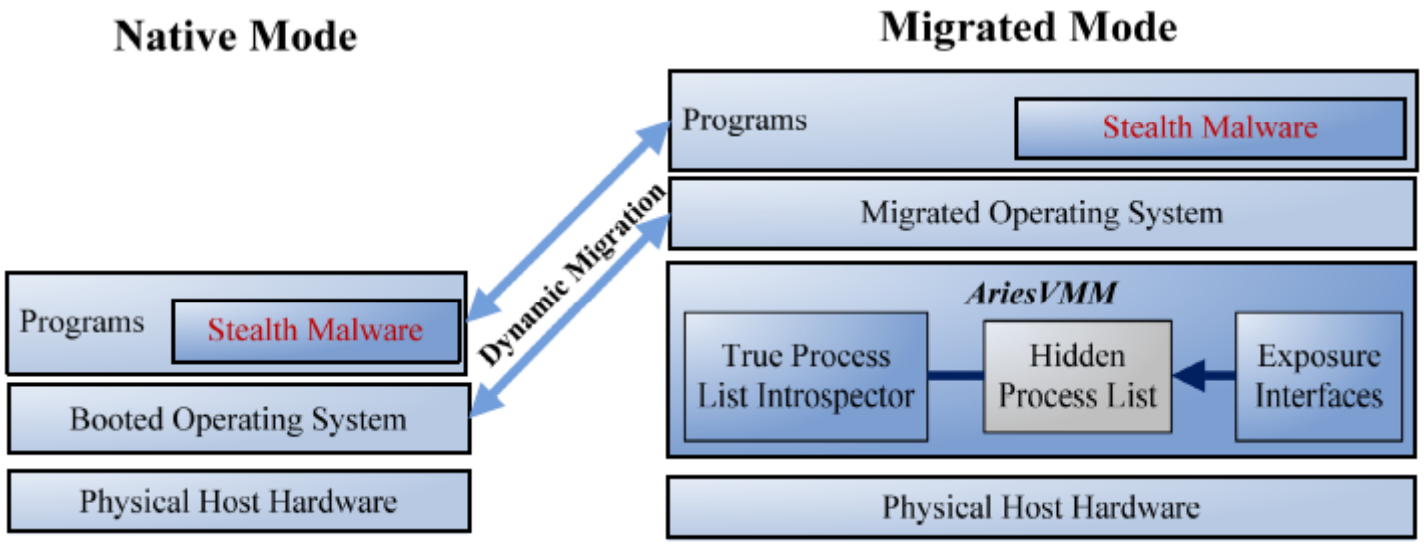
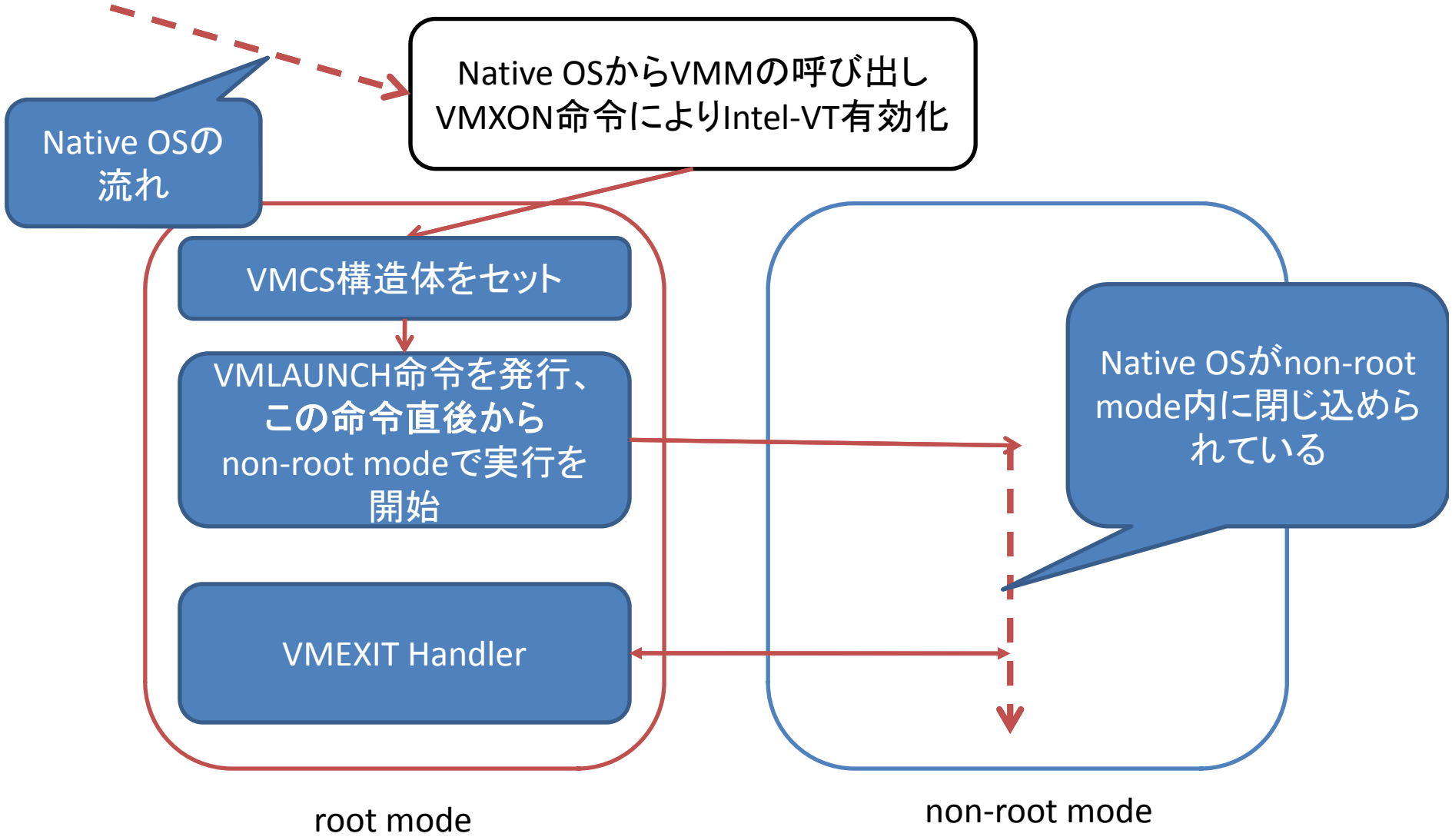
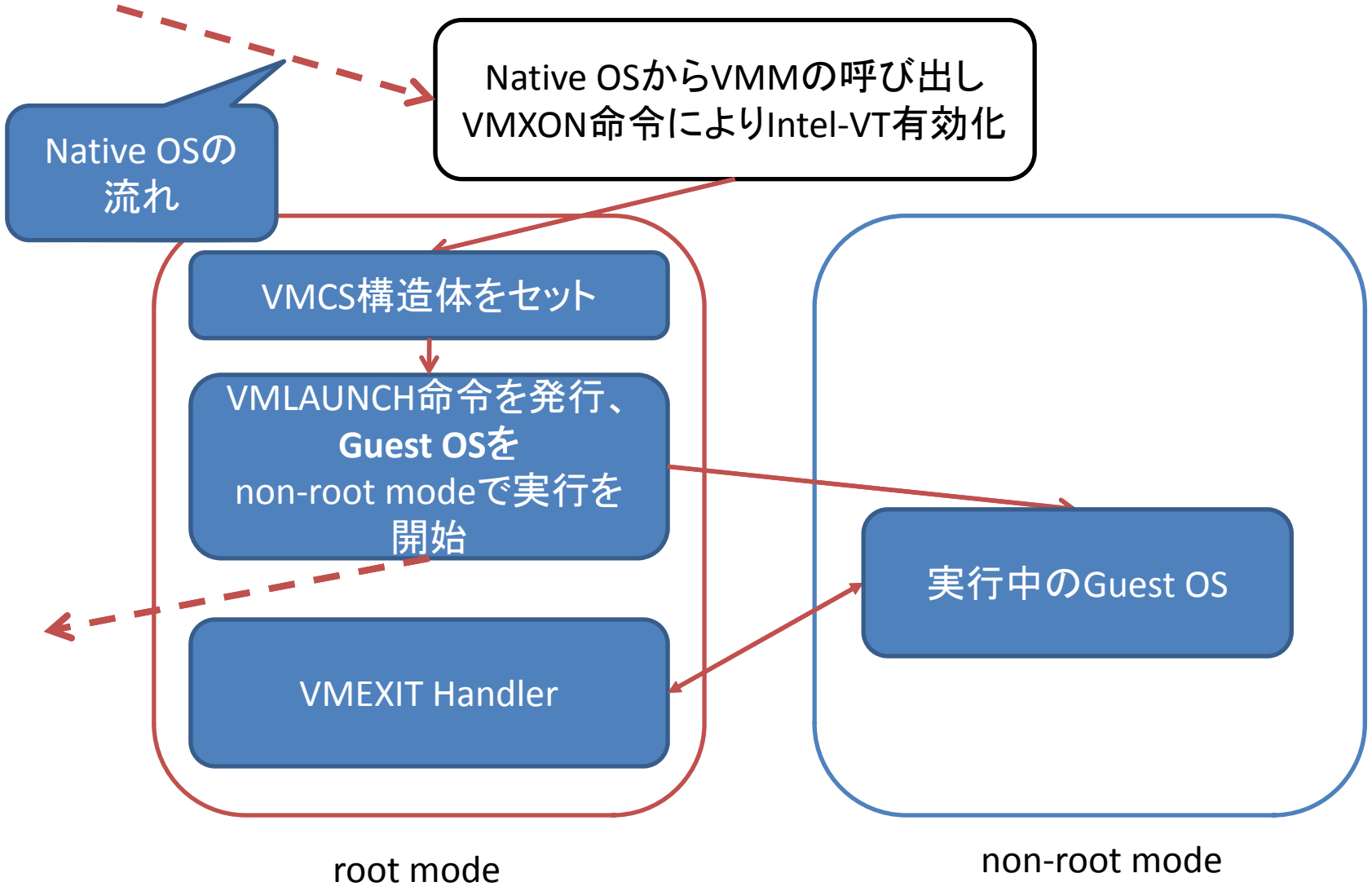



**Fig. 1.** The Architecture of Aries

# Dynamic Migration(3)



# 参考: 通常のVMM


Native OSからVMMの呼び出し
VMXON命令によりIntel-VT有効化
Native OSの
流れ
VMCS構造体をセット
VMLAUNCH命令を発行、
Guest OSを
non-root modeで実行を
開始
VMEXIT Handler
実行中のGuest OS
root mode
non-root mode

# Hidden Processの検出(1)

- Kernel-levelで取得できるProcess List
  - ZwSetSystemInformation()で得られるUntrusted Process Listを定期的に取得
  - Intervalはconfigurable, 1秒毎とした
- VMM-levelで取得できるProcess List
  - CR3レジスタに代入されるPage Directory Address(Address Space ID = Process Unique ID)
  - ASID->Process IDの変換も行う
    - ここがポイント(semantic gapの解決)
- 上記を比較することでHidden Processを検出

# Hidden Processの検出(2)

- VMM-level Process List
  - CR3レジスタへの代入は前述のMigrationによりTrap可能
  - Processの生成/終了はAntfarm[Jones, USENIX'06]に近い方法で取得する
    - 終了の検出が特徴的
    - 非ページのKernel-PageのPresent bitを用いて判定
    - 具体的にはEPROCESSと呼ばれるデータ構造が含まれるPageのPresent bitを監視している
  - 生成検出時にはOS側の情報を取得してCR3の値とProcess IDをひもづける
    - VMM自体が**Native OSの中に**存在しているので、
      - OS側の情報を取得すること
      - VMM-layerでの情報を取り出すこと
    - 両方可能

# Hidden Processの検出(3)

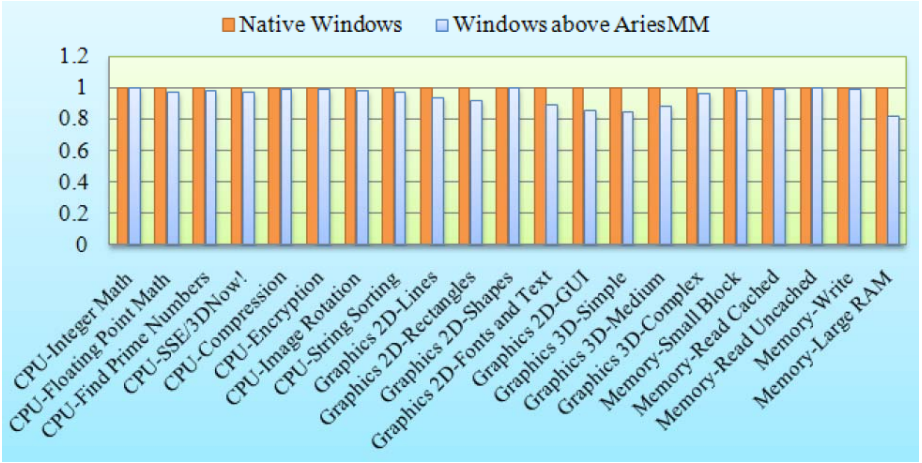
- VMM上で取得したTrue Process Listは特定のI/O portを経由してGuest OSに渡される
  - Guest OSが特定のportに対してI/O命令を発行
  - VMMが命令をTrap
    - Intel-VTの機能としてI/O port毎にTrapのON/OFFを切り替える機構がある(I/O bitmap)
  - 適切にデータを返す

# 評価(1)

- 環境
  - Core 2 Duo T7300(2.0GHz), 2GB Mem
  - Windows XP SP2
- 評価
  - 5種類のrootkitを感染させた場合にHidden Processが検出できるかどうか
    - 加えて既存のRootkit Detectorとの比較
    - 既存Detectorで検出できないものも全て検出できた
  - パフォーマンス
    - Windows版fork-wait、Benchmark tool(Passmark Benchmark 6.1)で評価

# 評価(2)

- fork-wait benchmark
  - VMMの挿入前後で15%のoverhead
  - 参考: VMWare, VirtualPC, Parallels上のXP: 19-23%のoverhead
- Benchmark tool
  - 平均でNativeの95.2%の速度で動作



# 関連研究

- Cross-view validation
  - Copilot[Petromi, USENIX'04] - user level
  - Strider GhostBuster[Wang, DSN'05] - kernel level
  - Livewire[Garfinkel, NDSS'03] - VMM level

- Libra[Yan, ISA'08]
  - Dynamic Migrationを用いないHidden Process Detection
- MiniVMM[Yan, SecureComm'08]
  - 同様の手法を用いてMigrationし、他VMMの特徴をEmulationすることで、**"VMM-aware malware"が動作しないようにする**

# まとめ

- Hardware-assist VMMとDynamic Migrationを用いてHidden Processの検出が行えることを示した
  - VMM-levelでの情報とOS-levelでの情報の両方を一括して取り扱うことでsemantic gapを解決
  - 必要最小限の仮想化によりOverheadは平均4.8%
  - 必要に応じてNative動作も可能