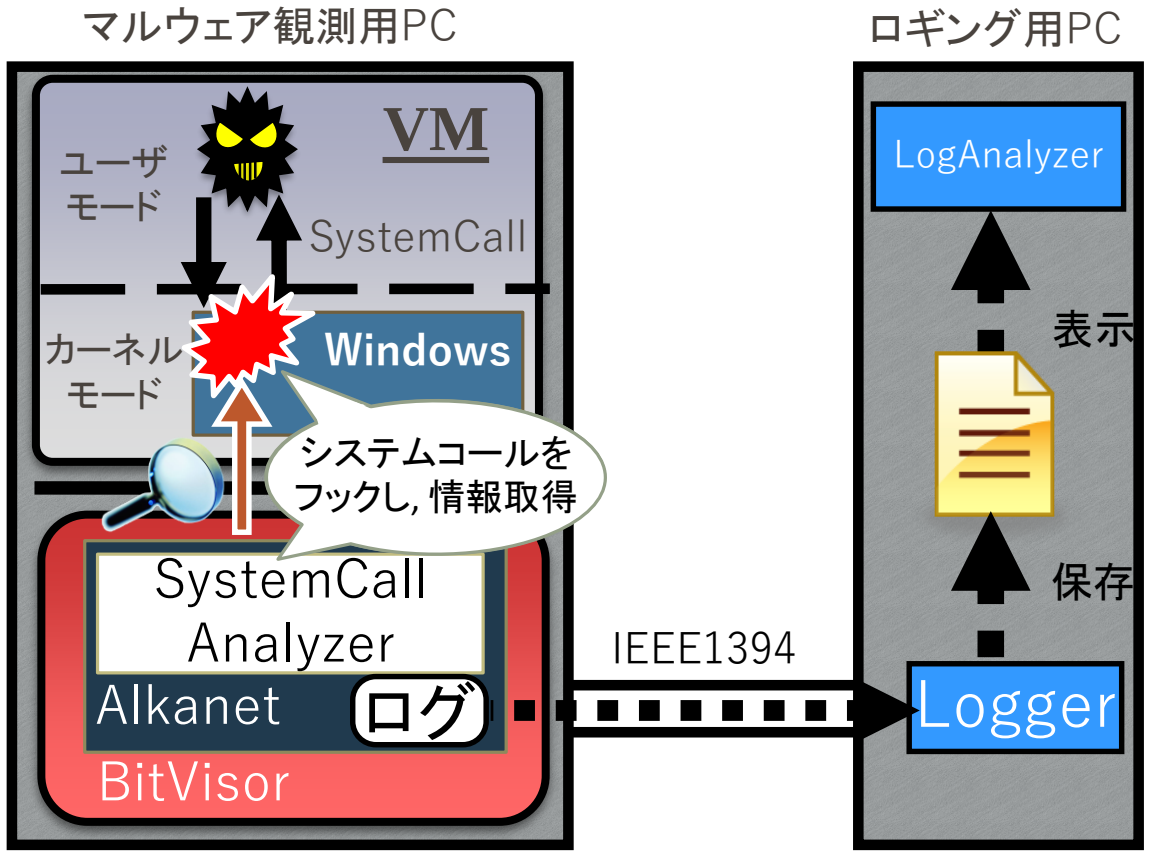
Virtualize APIC accessによるAPICフック手法

立命館大学
富田崇詠, 明田 修平 ,瀧本 栄二 ,毛利 公一

はじめに(1/2)

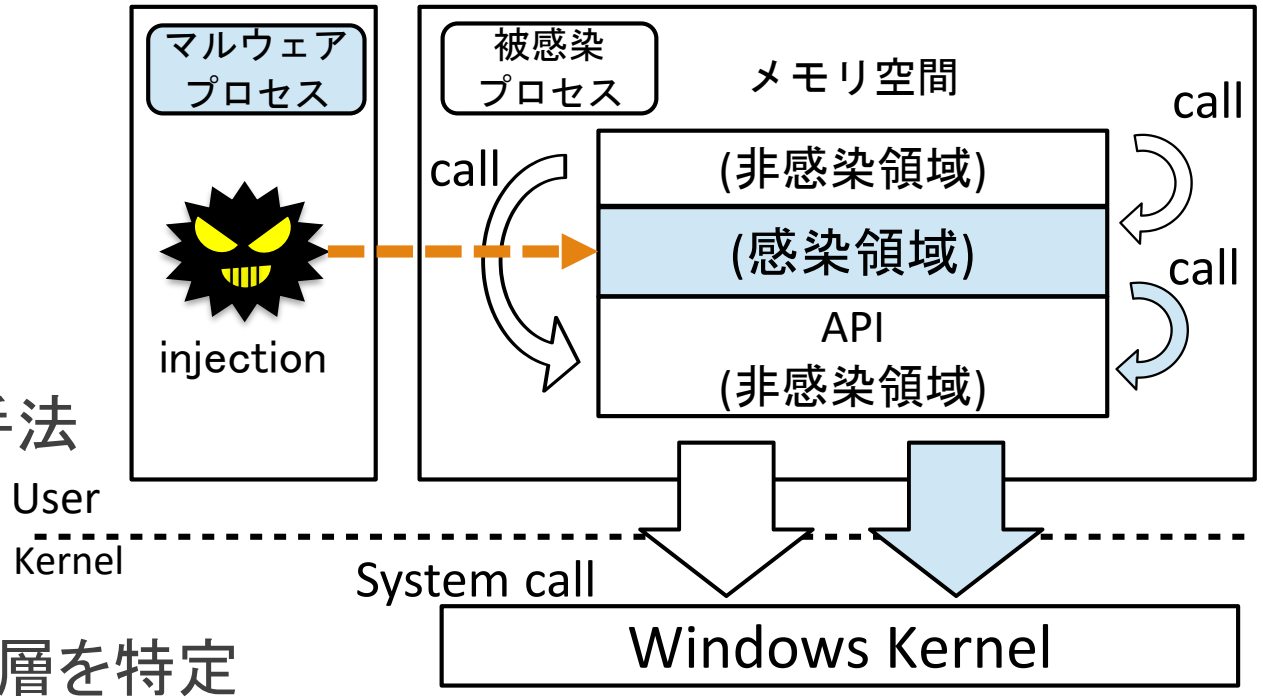
- マルウェアの脅威が問題となっている
  - 2015年に4億3000万以上の検体が新たに発見されている †
- マルウェア対策にはマルウェアが持つ機能・挙動の正確な解析が重要
- マルウェア動的解析システム： Alkanet
  - 仮想計算機モニタのBitVisorの拡張機能として動作
  - システムコールをプロセス・スレッド単位でトレースしマルウェアの挙動を観測



† Symantec: 2016年インターネットセキュリティ脅威レポート第21号 (2016).

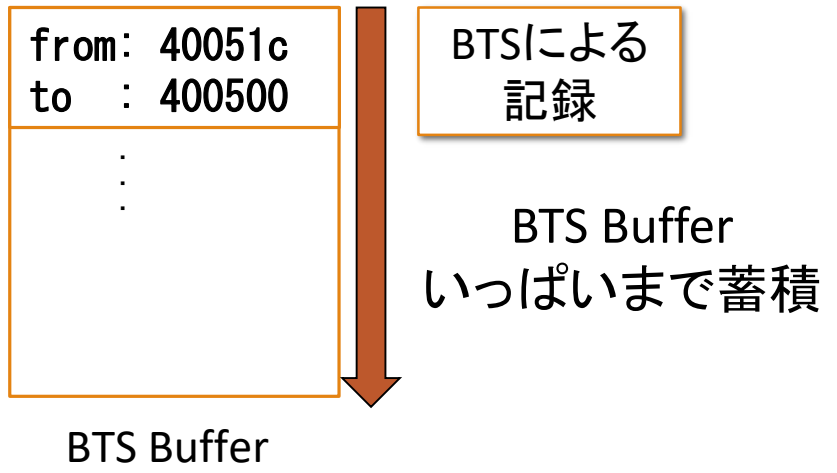
はじめに(2/2)

- 他プロセスへコード挿入を行うマルウェアが存在する
  - マルウェアに起因するシステムコールの識別が困難
- ブランチトレース機能を用いたシステムコール呼出し元アドレス取得手法
- Branch Trace Store(BTS)により取得した分岐元アドレス、分岐先アドレスからシステムコール発行時の関数呼出し階層を特定
- 発行されたシステムコールがマルウェアに起因するものか識別が可能



BTSを用いた分岐記録

- BTSの記録はBTS Buffer内に記録される
- バッファあふれが発生する場合がある
  - システムコール発行前の分岐を全て記録できない
  - マルウェアに起因するシステムコール発行を識別できない
- BTSの割込み機能を用いた継続的な分岐記録
  - BTS Bufferに限界まで記録が蓄積した時に割込みを発生させる
  - 割込み発生毎に記録を退避させる



BTSの割込み制御

- BTSの割込みを制御するには
  APIC内のLVT(Local Vector Table)を 変更する必要がある
  - 割込みベクタ番号・割込みのマスクなどの制御
- ゲストOSに変更されるとBTSの割込みが
  正常に動作しない可能性がある

APICへのアクセスをフックし
ゲストOSによる変更を防ぐ

BitVisor上でのAPICフック手法

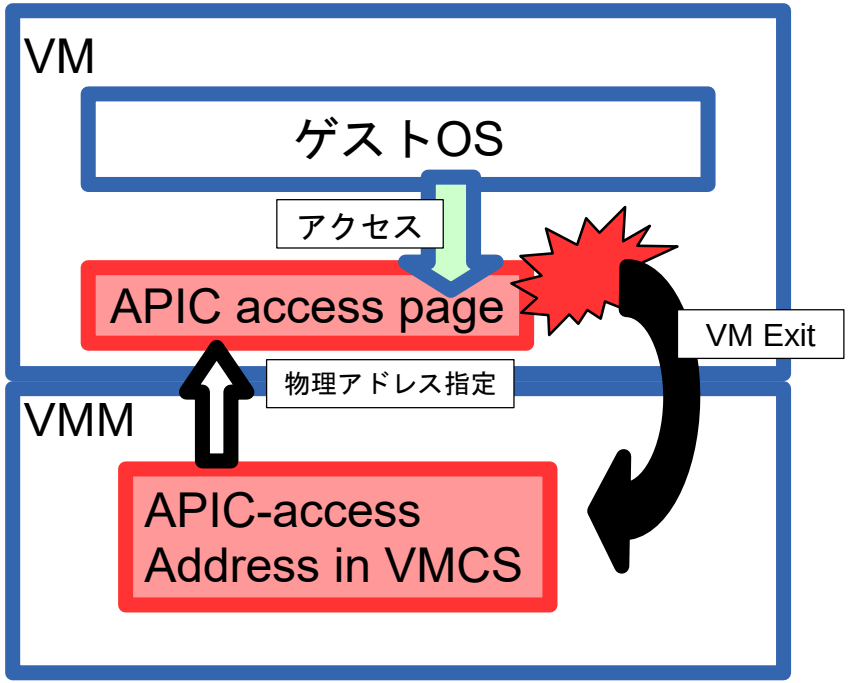
- BitVisor内の関数を用いたMMIOフック
  - EPT Violationを用いたフック手法
  - mmio_register関数でフック対象を指定できる
  - BitVisorで利用されている
- Intel VTのVirtualize APIC accessを用いたフック
  - APICページへのアクセス時にVM Exitを発生させる
  - BitVisorでは利用されていない

BitVisorでのMMIOフック

- EPT Violationによってフックを実現されている
  - Intel VT-xのアドレス空間仮想化機構
  - フック対象をマップしないことでEPT Violationを起こす
- APICのMMIO領域をフック対象に指定
  - 0xFEE0 0340(LVT PC)をフックする必要がある

Virtualize APIC accessを用いたフック

- VM Execution Controls
  - Virtualize APIC自体の有効化
- APIC access address
  - APIC pageの物理アドレスを指定
- APIC-access VM Exit(Exit Reason 44)時に実行される処理の追加
  - cpu_interpreter関数を実行



Virtualize APIC accessを用いた場合とMMIOフックを用いた場合の起動時間(1/2)

- BitVisor内の関数を用いたMMIOフック
  - LVTに限定してフックした場合
  - APIC全体をフックした場合
- Intel VTのVirtualize APIC accessによるフック

- これらの起動時間を確認
  - 各条件で3回ずつ計測
  - Windows XPで起動
  - GrubでのOS選択からデスクトップ画面表示までを計測

Virtualize APIC accessを用いた場合とMMIOフックを用いた場合の起動時間(2/2)

- BitVisor内の関数を用いたMMIOフック
  - LVTに限定してフックした場合
    - 平均4分6秒
  - APIC全体をフックした場合
    - 平均4分8秒
- Intel VTのVirtualize APIC accessによるフック
  - 平均4分10秒
- 起動時間に大きな差は出なかった

フック手法による比較

- 両手法とも起動時間に大きな差はなかった
  - 両方APICページ全体をフックしているためか
- BitVisorのMMIOフック機能
  - EPT Violationを応用した機能のため，
    これ以上のオーバヘッド削減方法がなさそう
- Intel VT-xのVirtualize APIC access
  - VM Exit回数削減のための機能がある

オーバヘッド削減の余地がある

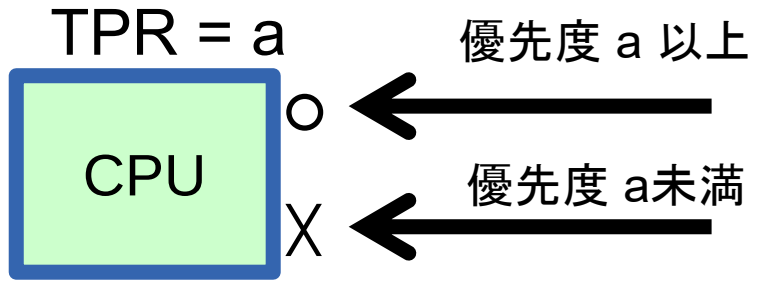
Virtualize APIC accessを利用

- オーバヘッド削減のための機能が存在する Virtualize APIC accessを採用
- APICレジスタへのアクセスを仮想化しVM Exit回数の削減
  - TPR Shadow
    - TPRへのアクセスを仮想化
  - APIC Register Virtualization
    - TPR以外のレジスタ(EOI,IRR,ISRなど)を仮想化
    - フックが必要なLVTへアクセスした時のVM Exitも削減されてしまう

TPR Shadowを用いて動作を確認する

TPR

- TPRの値と割込みの優先度を比較し，マスクするか判断される
- 割込みの優先度はベクタ番号から決定される
  - 割込み優先度 = ベクタ番号/16
- Windowsの起動時に発生するAPICアクセスのほとんどがTPR

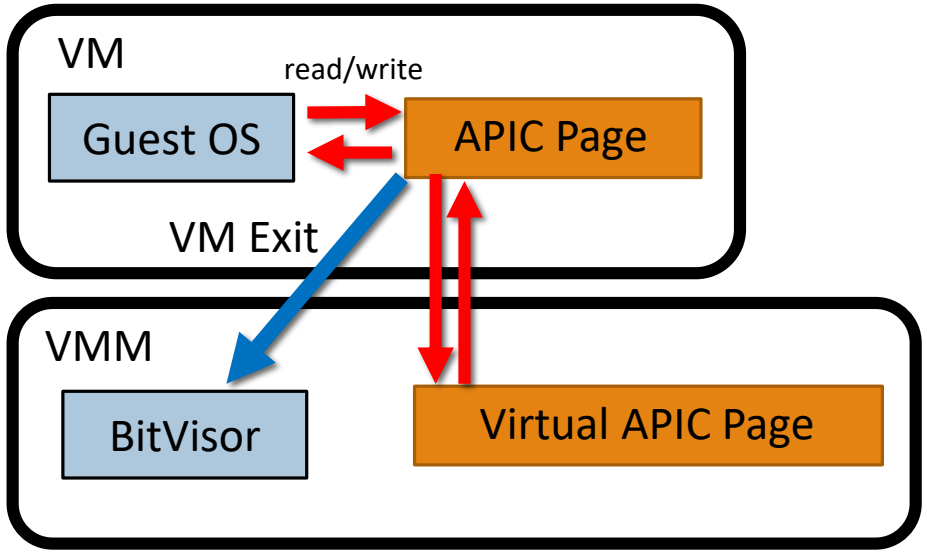


TPR Shadowを用いたVM Exit削減

- TPRレジスタへのアクセスを仮想化
- VMCS内のVM Execution Controlsのビットを変更し有効化
- その他に必要な設定
  - Virtual APIC page
  - TPR Threshold

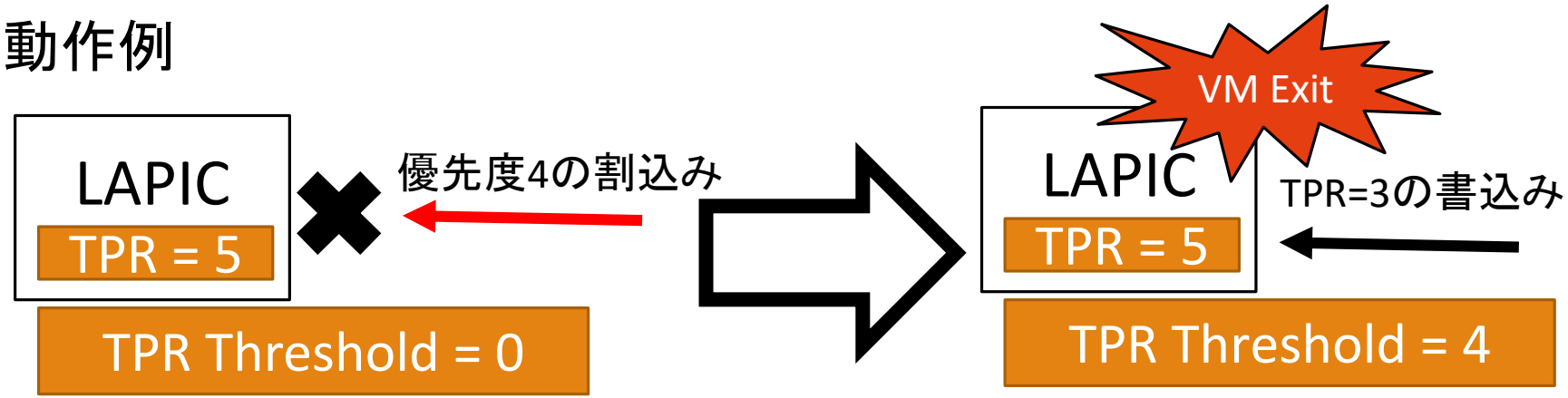
その他の必要な設定

- Virtual APIC pageの設定
  - メモリを確保し、その物理アドレスを指定
  - TPRの値を格納する場所として用意
- TPR Threshold
  - TPRへの書込み時に，
    VM Exitを起こすかどうかの閾値
  - 書込まれたTPRの値が
    TPR Threshold以下ならVM Exit



TPR Thresholdの変更(1/2)

- Xenのソースコードを参考に処理を追加
- TPR Thresholdの変更
  - 割込みがブロックされた時に値を変更する
  - ブロックされた割込みと同じ優先度を設定する



TPR Thresholdの変更(2/2)

- IRR(割込み要求レジスタ)，ISR(インサービスレジスタ)から APICが受信している割込みベクタ番号を取得できる
- 追加した関数
  - check_tprblock
    以下2つの関数を実行し，その結果からTPR Thresholdを変更する
  - vlapic_has_pending_irq
    ISR,IRRの値をread
  - interrupt_blocked
    TPRの値とISR,IRRの値を用いてブロックの原因を識別

TPR Shadow動作検証(1/2)

- Virtualize APIC access，TPR Shadowを有効にし起動
- TPRアクセスによるVM Exit発生時にログ出力
  - TPRによるVM Exitが削減されているかを確認するため
- 正常に起動するか確認
- ゲストOSとしてWindows XPを実行

TPR Shadow動作実験(2/2)

- TPR アクセスによるVM Exit時のログは出なかった
  - VM Exitの削減に成功している
  - TPR Shadow自体は動作している
- ロゴ画面が表示された瞬間に停止
  - 起動までは確認できず
- 動作停止の原因は調査中

おわりに

- Virtualize APIC accessを利用したAPICフックを確認
- MMIOフックとVirtualize APIC accessで
  起動時間に大きな差はなかった
- Virtualize APIC accessには
  VM Exit回数を削減できる機能がある
  - TPR Shadowを試したが起動中に停止
- 今後の予定
  - 動作停止について調査
  - TPR Shadowを用いた場合の起動時間計測