Monthly Research

VirtualBoxを利用したマルウェア実行時検査の自動化

株式会社ＦＦＲＩ
http://www.ffri.jp

Ver2.00.01

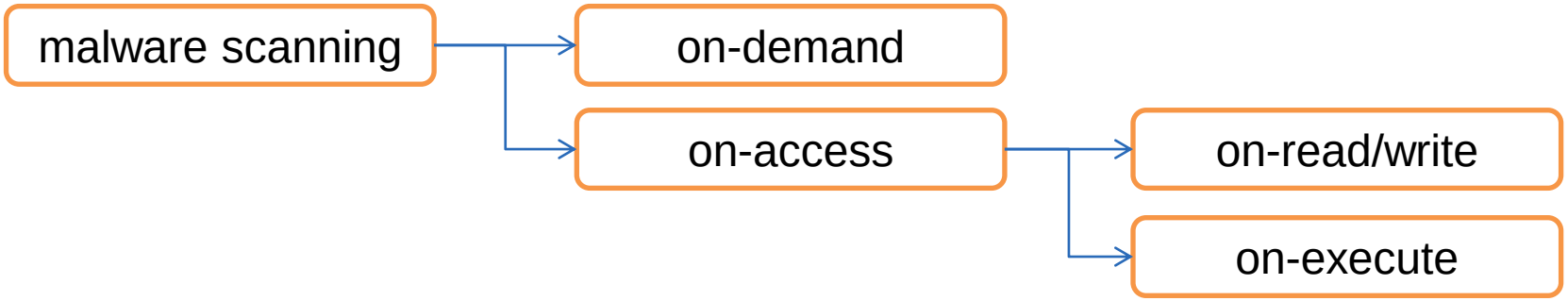
Agenda

1. 背景と目的
2. 検査の概要
   - 自動的な実行時検査
   - 仮想化ソフトウェアと自動化手法
   - Oracle VM VirtualBoxとその自動化手法
   - VBoxManageの使用例
3. 自動化スクリプト
   - FFRI AutoMonkey
   - 設計コンセプト
   - スループット
   - パフォーマンス
4. 参考情報
5. フィードバック

1.背景と目的

- マルウェア検知エンジンの評価等において大量マルウェアの自動検査が必要
- 検査方式はon-demand(ユーザー要求時), on-access(アクセス発生時)に大分される
- 中でもプログラム実行時(on-execute)の検知はマルウェアを1件ずつ実行する必要がありファイル一括での検査が困難
- そのため仮想化技術等を利用した自動化が必要
- 本書では、Oracle VM VirtualBoxを利用した実行時検査の自動化手法を紹介

■マルウェア検査方式の種類



2.1.自動的な実行時検査

- 基本的には下記のステップで1件ずつマルウェアを実行し、検査結果を得る
  1. マルウェアを1件ずつゲストにコピー
  2. 当該マルウェアをWindowsゲスト内で一定時間実行
  3. ゲスト内に設置した仕組みにより検知、分析等を行う
  4. 実行終了または一定時間経過後、検知・分析結果を保全する
  5. ゲストの環境をマルウェア実行前に復元する
  6. 1に戻る

- 上記より実現に必要な機能は以下の通り
  a. ゲストにファイルをコピーする機能(copy-to)
  b. ゲスト内のプログラムを実行する機能(exec)
  c. ゲストからファイルをコピーする機能(copy-from)
  d. ゲストの状態を復元する機能(revert)

  → いずれの機能もホスト-ゲスト間でTCP/IP等で通信することで実現可能
  但し、今回は可能な限りこれら作り込みをしない方式を模索

2.2.仮想化ソフトウェアと自動化手法

- 仮想化ソフトウェアが標準的(一部 3rd party)に備えている仕組みを利用
- VMware(有償)およびVirtualBoxは必要な機能を完備
  → 本書ではコストメリットからVirtualBoxを利用した方法を検討
- QEMU+KVMは3rd partyソフトウェアで実現可能 (例: libguestfs + winexe)
  - “Malware Analysis: Collaboration, Automation & Tuning”, Shmoocon 2013
    http://www.slideshare.net/xabean/malware-analysis-16674048

ソフトウェア	ライセンス	copy-to	copy-from	exec	revert	手法
VMware Workstation	Proprietary	○	○	○	○	VIX API
VMware ESX(#1)	Proprietary	○	○	○	○	VIX API
Oracle VM VirtualBox	GPL2	○	○	○	○	VBoxManage
QEMU + KVM	GPL2(#2)	×	×	×	○	Libvirt

#1 ESXiは、評価ライセンスを利用することで60日間VIX APIを利用可能
#2 QEMUはGPL2, KVMはコンポーネント毎にGPL/GPL2/LGPL等

2.3.Oracle VM VirtualBoxとその自動化手法

- x86仮想化ソフトウェアのひとつ、現在はオラクルが開発
- バージョン4.0以降、完全なオープンソースソフトウェアとして提供
- 比較的広範囲のホスト/ゲストOSをサポート
  - ホストOS：Windows, Linux, Mac OS X, Solaris
  - ゲストOS：Windows, Linux, FreeBSD, OpenBSD, Mac OS X Server, Solaris等
- VBoxManageによるCLIベースの自動化が可能
  - guestの起動、サスペンド、レジューム、電源断、クローニング、状態確認
  - guestからのファイルコピー、guestへのファイルコピー
  - guest内プログラムの実行
  - スナップショットの取得、復元
  - 仮想マシンのハードウェア制御、等

2.4.VBoxManageの使用例

ゲストの起動

```
% vboxmanage  startvm   vm
```

ゲストの停止

```
% vboxmanage  controlvm  vm  poweroff
```

スナップショットの復元

```
% vboxmanage  snapshot  vm  restore  snapshot-1
```

ゲスト内プログラムの実行

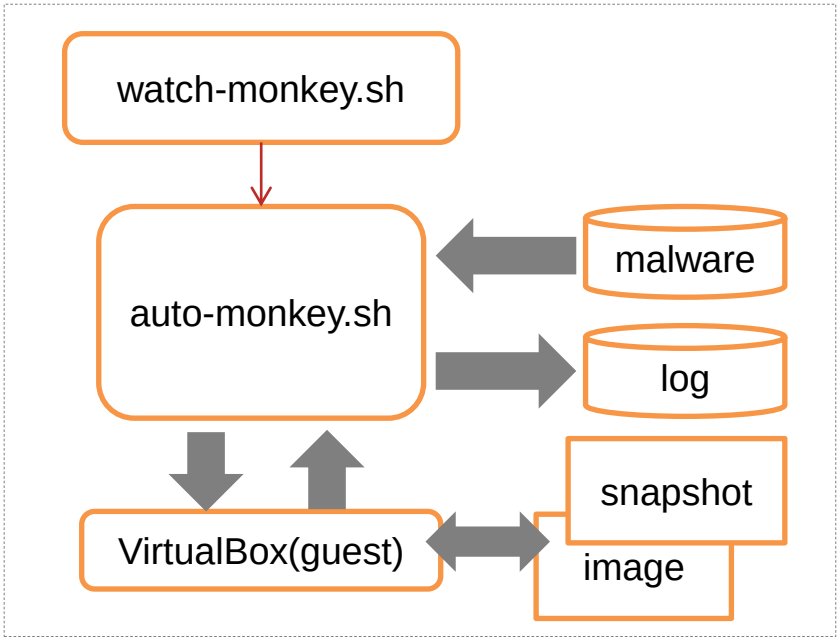
```
% vboxmanage  guestcontrol  exec  vm  --image “c:/windows/system32/calc.exe” ¥
    --username admin --timeout  60000  --wait-exit
```

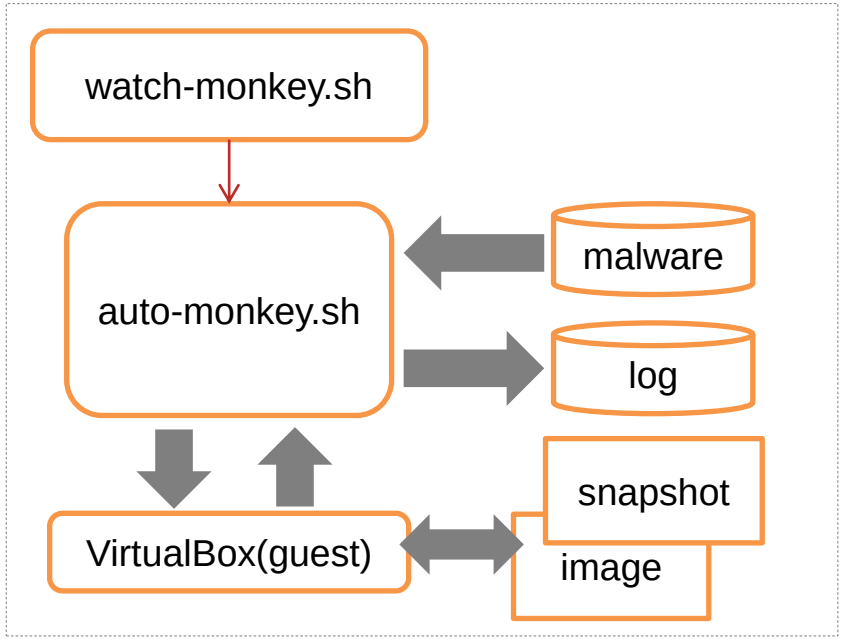
ゲストへのファイルコピー

```
% vboxmanage  guestcontrol  vm  copyto  “/some/file”  “c:/file.txt”  --username admin
```

3.1.FFRI AutoMonkey

- VBoxManageを利用した自動化スクリプト(bashスクリプト)
  - auto-monkey.sh: 検体のコピー、実行、ログ取得、イメージ復元を自動化
  - watch-monkey.sh：auto-monkey.shを監視、停止時に処理を再開
- 複数同時実行も可能、各スクリプトはそれぞれ個別に動作
- 下記にて配布、使い方はREADMEを参照（3条項BSDライセンス）
  - http://www.ffri.jp/research/freeware.htm





3.2.コンセプト

- “シンプルに動作し続ける”
- 大量のマルウェアを処理する上で残処理時間を予測することは重要
  - ハングアップしたまま停止するといつ終わるか分からない
- VBoxManageやVIX API等の安定性はこれら自動化の生命線
- 但し、多くの場合長時間実行し続けるとエラーが発生する
  - エラーによる処理の失敗
    - 即座にスクリプトの実行を中止
    - watch-monkey.shにより自動的に再開
  - エラーによるハングアップ（stuck)
    - watch-monkey.shがハングアップを監視し、関連プロセスを終了・再開

3.3.スループット

- 1ホスト、7ゲストで7セッション同時実行

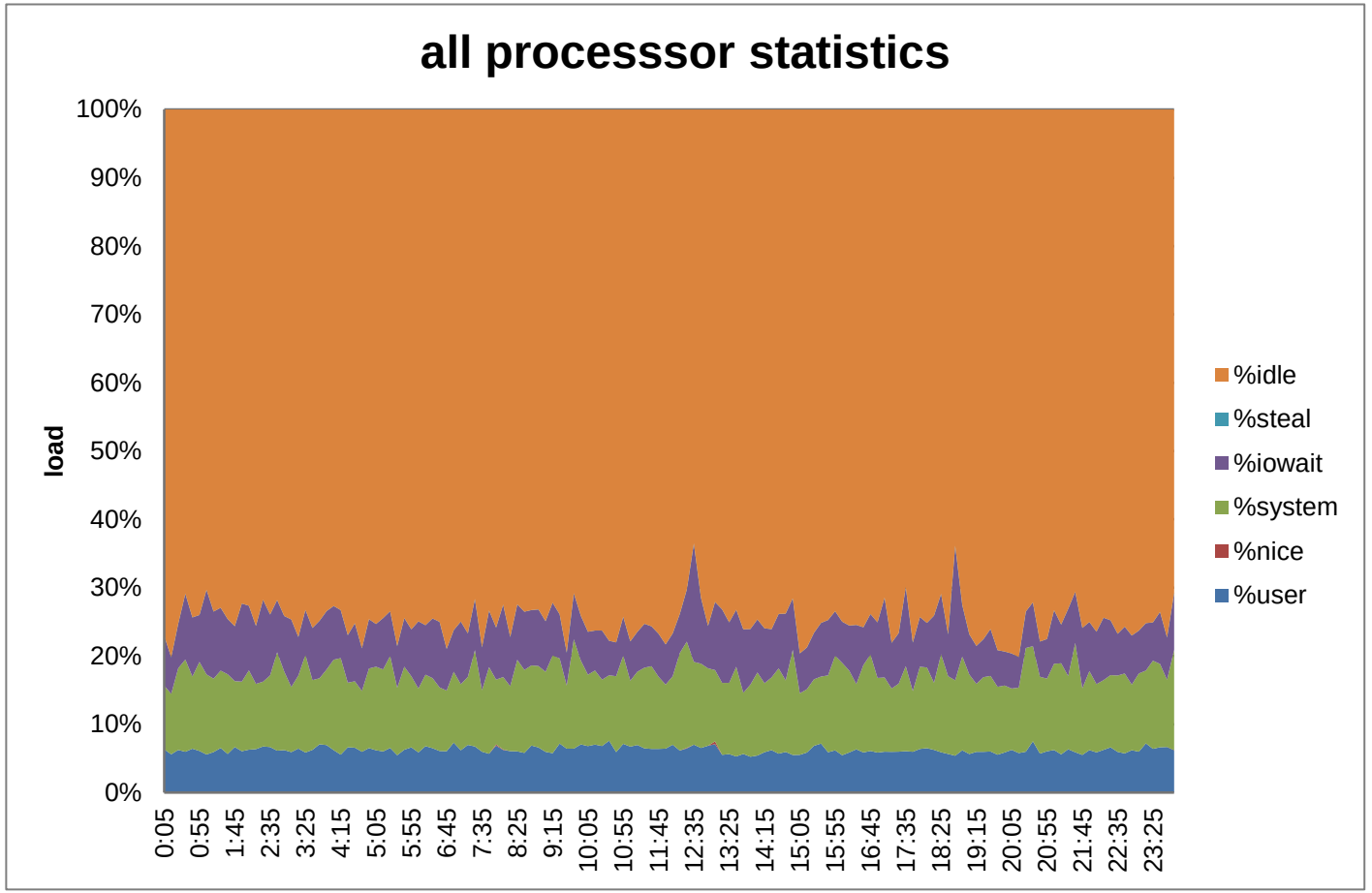
- 20,000件のマルウェアを1件60秒間実行
  - 所要時間：37時間15分
  - スループット：8.95件/分
    ※60秒以内にマルウェアの実行が完了した場合、直ちに次マルウェアを処理するためこれらの値は処理対象となるマルウェアによって変動

- ホスト及びゲスト環境は下表の通り

ハードウェア	CPU: Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz Memory: 8GB HDD: 1.8TB x 1
ホストOS	Ubuntu 13.04 + VirtualBox 4.2
ゲストOS	Windows XP SP3(x86) + FFR yarai 2.3 CPU:1 CPU Memory:750MB

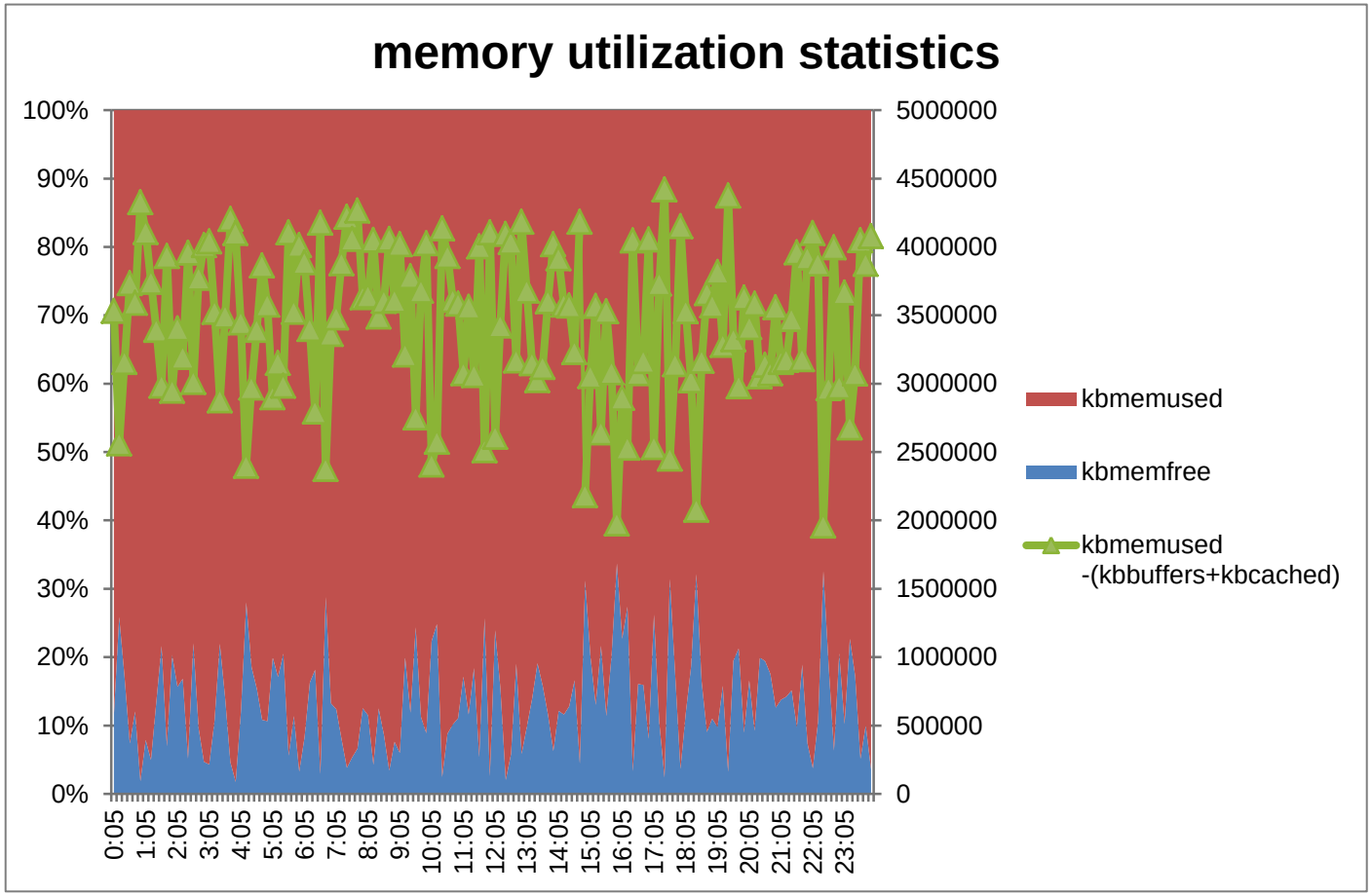
3.4.パフォーマンス – プロセッサ

- プロセッサ全体で常時70%程度がアイドル状態（各コア単体でも同様の傾向）



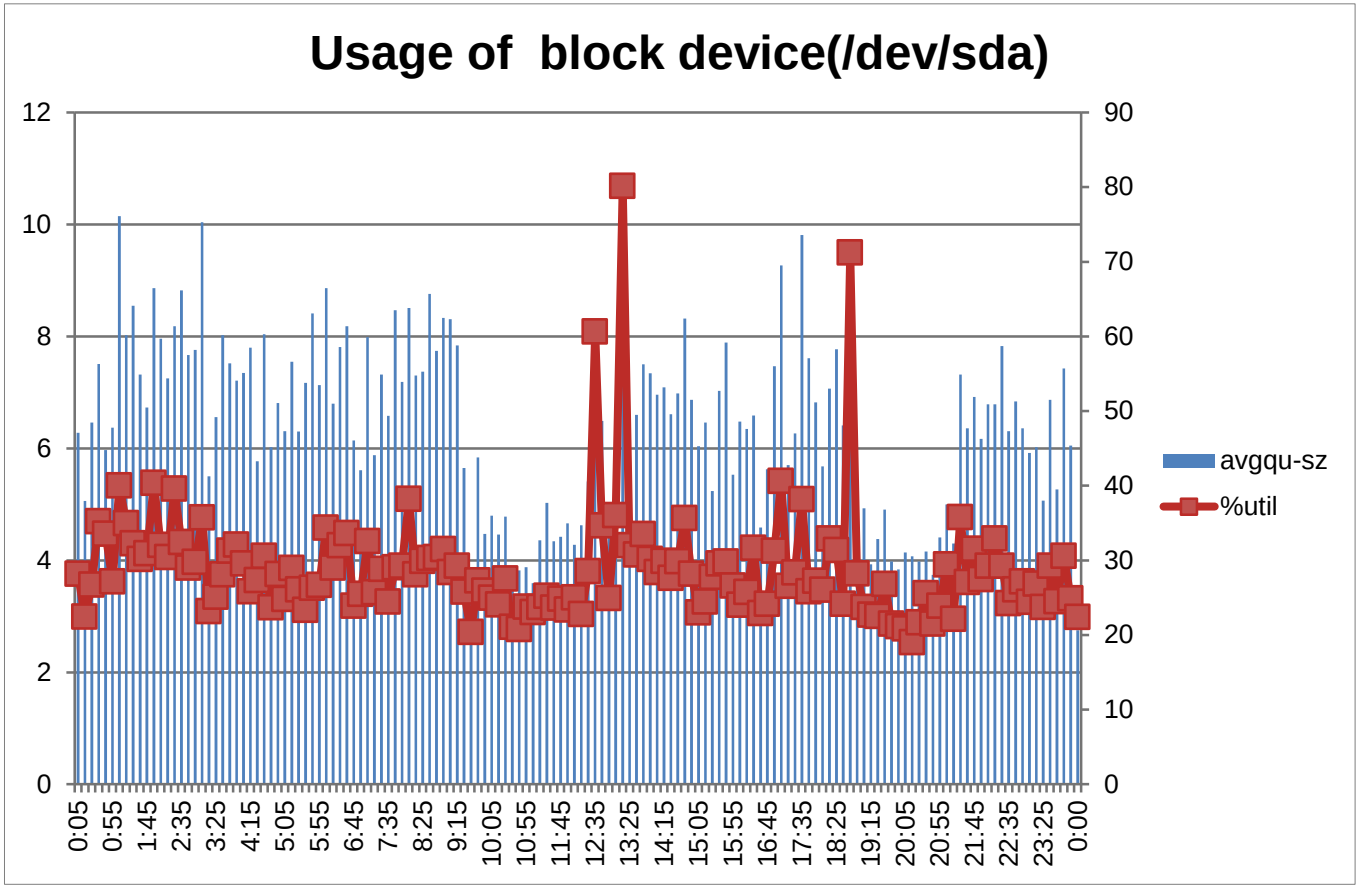
3.4.パフォーマンス – メモリ

- 常時80～90%程度のメモリを消費（但し、実メモリ使用量は2.5GB～4.0GBを推移)



3.4.パフォーマンス - Disk IO

- ディスクビジー率(%util)は、おおよそ30%で安定
- キューイングされたリクエスト数も4～10件で安定



3.4.パフォーマンス – 考察

- 1ホスト7ゲストの環境でCPU, メモリ, IOいずれも大きなボトルネックにならずに連続動作可能
- IOに若干の懸念が想定されるが10ゲスト程度までスループットを通さずに同時実行可能と考えられる（メモリ使用率より）
- 但し、ゲスト内で実行する処理次第で要件が異なるため注意が必要

4.参考情報

- http://www.ffri.jp/assets/files/research/freeware/FFRIAutoMonkey-1.0.tgz
- https://www.virtualbox.org/manual/UserManual.html
- http://www.slideshare.net/xabean/malware-analysis-16674048
- http://www.youtube.com/watch?v=peHdyUlchSM
- http://libguestfs.org/
- http://sourceforge.net/projects/winexe/files/

5.フィードバック

- E-Mail
  - research-feedback@ffri.jp
- Twitter
  - @FFRI_Research