KVMIveggur: Flexible, secure, and efficient support for self-service virtual machine introspection

By:
Stewart Sentanoe (University of Passau), Thomas Dangl (University of Passau), and Hans P. Reiser (University of Passau)

From the proceedings of

The Digital Forensic Research Conference

DFRWS USA 2022

July 11-14, 2022

KVMIveggur: Flexible, secure, and efficient support for self-service virtual machine introspection

Stewart Sentanoe, Thomas Dangl, Hans P. Reiser
se@sec.uni-passau.de
University of Passau, Germany
DFRWS USA - 12.07.2022

Funded by
Deutsche Forschungsgemeinschaft
German Research Foundation

SPONSORED BY THE
Federal Ministry of Education and Research

Outline

1. Introduction
2. Background
3. Related Works
4. *KVMIveggur* Design
5. Implementation
6. Evaluation & Discussion
7. Conclusions

Introduction

Introduction

- Virtual machine introspection (VMI)
  ➜ monitoring a virtual machine (VM) from the outside
  ➜ gain information about inner state
- VMI requires interaction with the hypervisor (sensitive)
- VMI has caught traction by hypervisor and anti-virus vendors (VMware, Kaspersky Lab, BitDefender and GData)
- IaaS (Infrastructure-as-a-service) is becoming more popular
- Lack of VMI implementation on public cloud providers
- Our research is to create access control of VMI

Background

Virtual Machine Introspection

- Analyze a guest VM from the outside with help of the hypervisor
- Requires no guest agent
- Two methods:
  - Asynchronous or passive
  - Synchronous or active
- Xen ➔ hypercalls (privileged) ➔ Dom0
- KVM ➔ UNIX domain socket
- Use-cases: malware analysis, deception technology, OS hardening, etc
- Publicly available VMI libraries
  - LibVMI
  - LibKVMi
  - Drakvuf
  - Libvmtrace

Related Works

Current SoTA

- CloudPhylactor → leverage Xen security modules [1]
- TwinPorter → extension of CloudPhylactor → live migration (Xen) [2]
- CloudVMI → expose VMI capabilities over RPC [3]
- Furnace → sand-box the VMI application (with SELinux and Seccomp-BPF) [4]
- FROST → based on OpenStack, predefined digital forensic tools [5]

[1] Taubmann, B., Rakotondravony, N. and Reiser, H.P., 2016, August. Cloudphylactor: Harnessing mandatory access control for virtual machine introspection in cloud data centers. In 2016 IEEE Trustcom/BigDataSE/ISPA (pp. 957-964). IEEE.

[2] Taubmann, B., Böhm, A. and Reiser, H.P., 2019, August. TwinPorter-An Architecture For Enabling the Live Migration of VMI-Based Monitored Virtual Machines. In 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) (pp. 427-434). IEEE.

[3] wook Baek, H., Srivastava, A. and Van der Merwe, J., 2014, March. Cloudvmi: Virtual machine introspection as a cloud service. In 2014 IEEE International Conference on Cloud Engineering (pp. 153-158). IEEE.

[4] Bushouse, M. and Reeves, D., 2018, September. Furnace: Self-service tenant vmi for the cloud. In International Symposium on Research in Attacks, Intrusions, and Defenses (pp. 647-669). Springer, Cham.

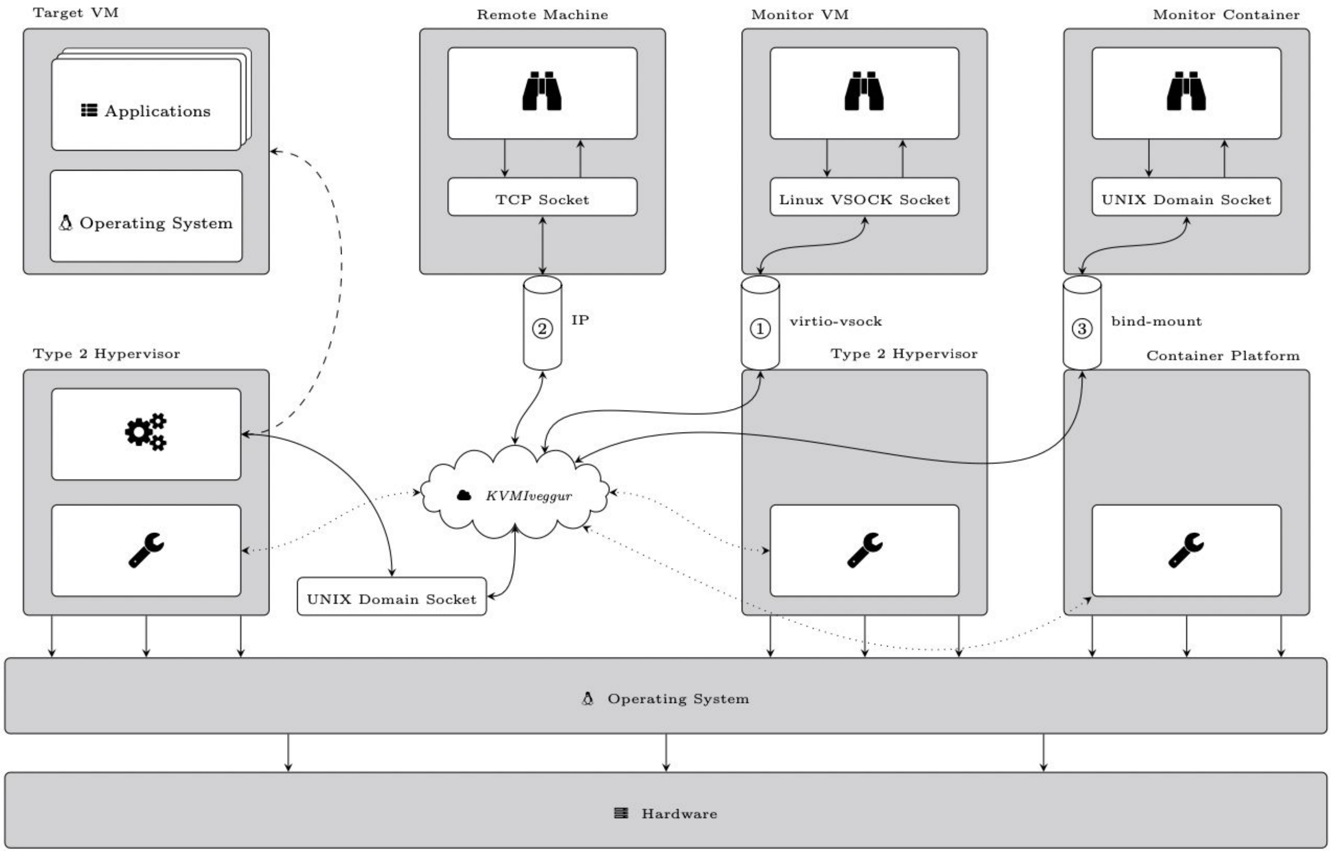
[5] Dykstra, J. and Sherman, A.T., 2013. Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. Digit. Invest. S87–S95.

KVMIveggur Design

Goals & Assumptions

- Goals
  - Multiple flavors
  - Self-service
  - Secure access
  - Isolation
  - Easy integration
- Assumptions
  - The cloud provider is trusted
  - The operating system is trusted
- Security
  - Access to the monitor system is protected (e.g. via SSH)
  - Unauthorized access will harm only the monitored VM

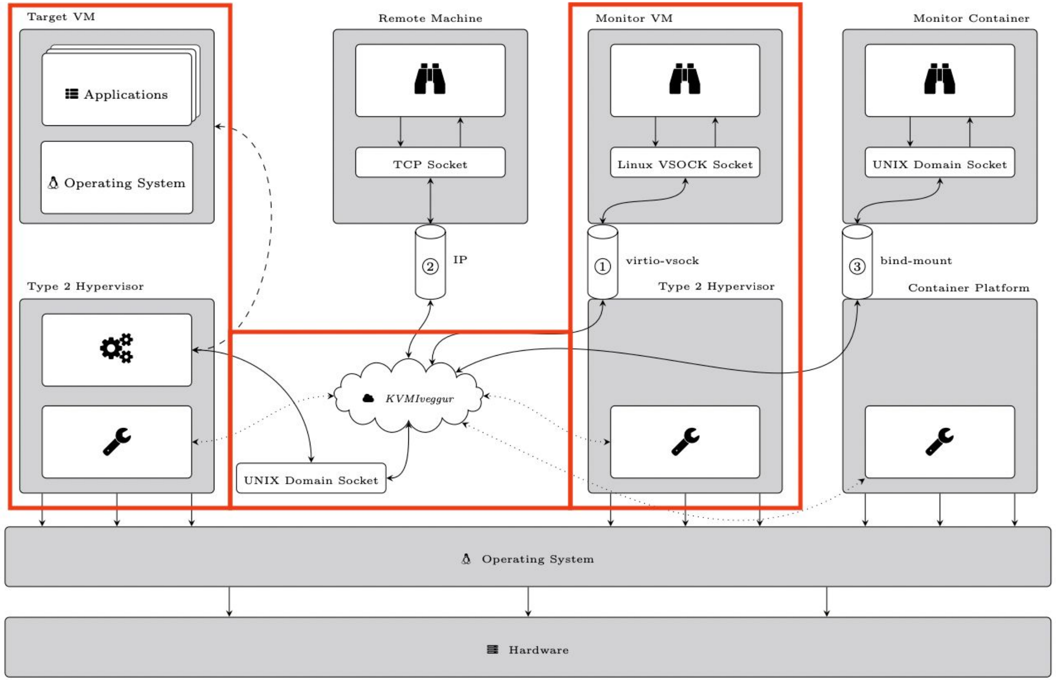
Overview



Target VM
Applications
Operating System
Remote Machine
TCP Socket
Monitor VM
Linux VSOCK Socket
Monitor Container
UNIX Domain Socket
2
IP
1
virtio-vsock
3
bind-mount
Type 2 Hypervisor
Type 2 Hypervisor
Container Platform
KVMIveggur
UNIX Domain Socket
Operating System
Hardware

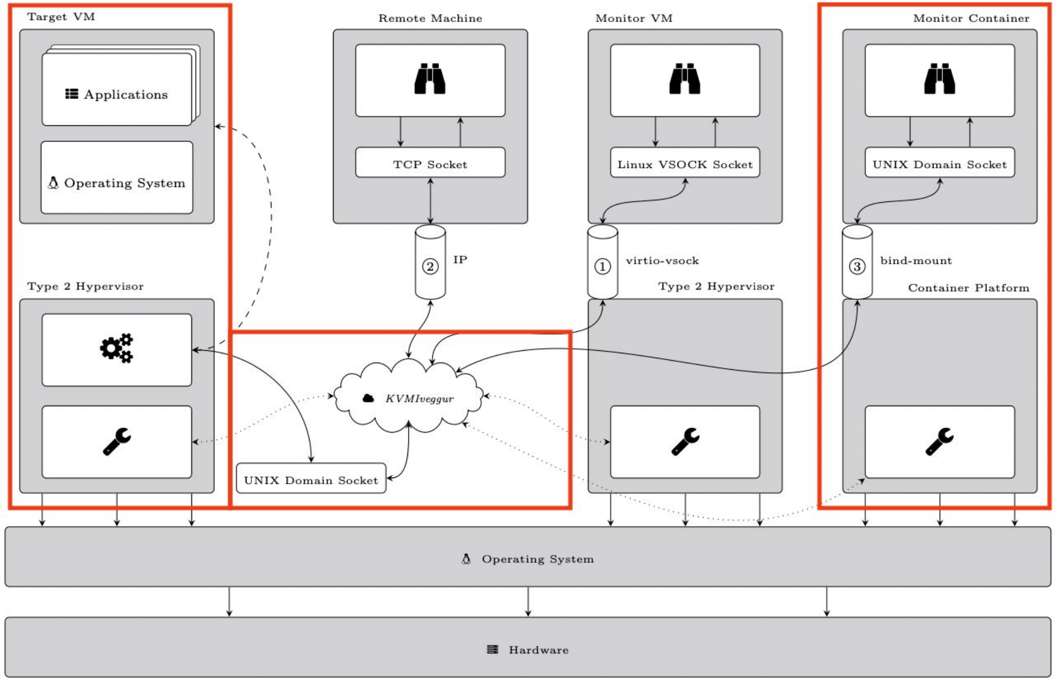
VM-to-VM

- Leverage *virtio-vsock*
- Expose the socket on the hypervisor into the monitor VM



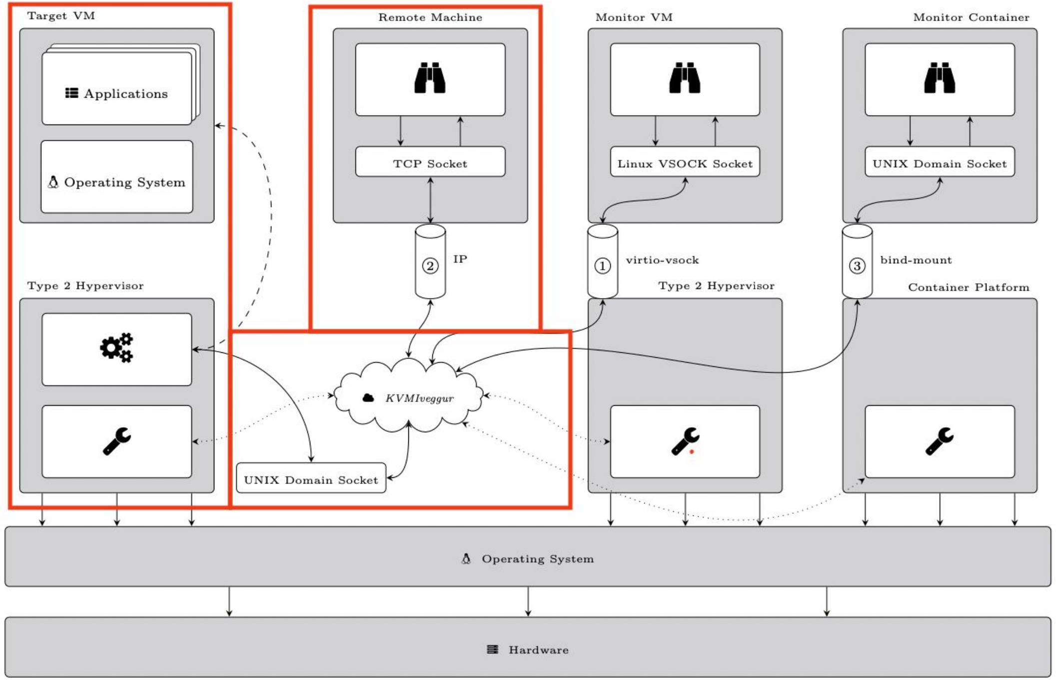
VM-to-Container

- Leverage *bind-mounts*
- Expose the socket on the hypervisor into the monitor Container



Over-the-Network

- Leverage *socat*
- Expose the socket on the hypervisor via the network
- For the current version, not encryption is used



Implementation

On Hardware and Paravirtualized Machine

- On the monitoring VM

```
<vsock model="virtio">
    <cid auto="no" address="{monitoring cid}"/>
    <alias name="{vsock name}"/>
    <address type="pci" domain="0x0000" bus="0x00" slot="0x0b" function="0x0"/>
</vsock>
```

On Hardware and Paravirtualized Machine

- On the target VM

```
<qemu:commandline>
  <qemu:arg value="-chardev"/>
  <qemu:arg value="socket,cid={monitoring cid},port={vsock port},id=chardev0,reconnect=10"/>
  <qemu:arg value="-object"/>
  <qemu:arg value="introspection,id=kvmi,chardev=chardev0"/>
</qemu:commandline>
```

Over the Network

- Transform file-based socket to TCP stream

$ socat UNIX-LISTEN:**{target VM's UDS location}**,unlink-early,fork TCP:**{client's IP address}**:**{target port}**,fork,end-close

- Transform back the TCP stream to file-based socket

$ socat TCP-LISTEN:**{target port}**,reuseaddr,reuseport,fork UNIX-CONNECT:**{UDS new location}**

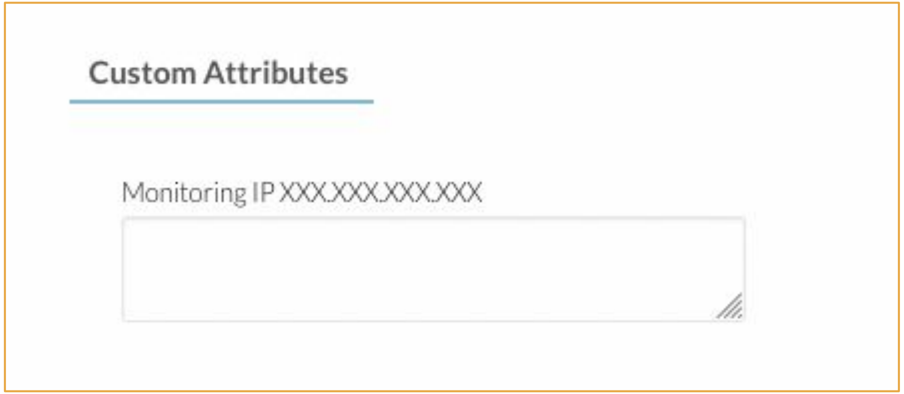
On Docker Container

- Mount the folder that contains the socket to the container

```
$ docker run <related container options> -v <UDS location on the host >:<UDS location on the container >
```

OpenNebula Integration

- Add custom hook script (written in Python)
- Add additional fields on the user interface (using custom attributes)



Evaluation & Discussion

Performance

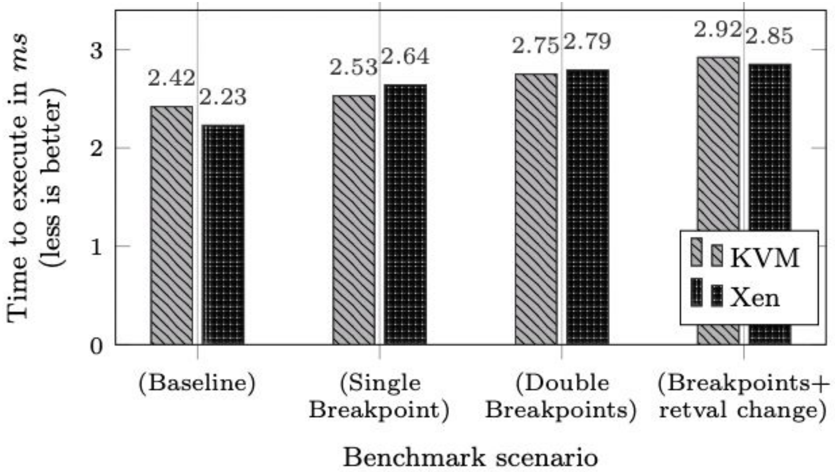
- Legend
  - *NoVMI* → baseline
  - *Native* → native VMI application runs directly on the host OS
  - *Virtio* → VMI application runs on different VM (monitor VM)
  - *NetCoHost* → VMI application over the network, but located on the same host
  - *NetRemote* → VMI application over the network (remote client)
  - *FromDocker* → VMI application runs inside a docker container
- Benchmark environment
  - Intel Xeon E3-1230 v5 CPU at 3.40 Ghz (hyper-threading enabled) – main
  - Intel Xeon E5-2609 v3 CPU at 1.90 Ghz – as migration target
  - 64 GB of RAM
  - Debian 11 (kernel 5.4.24)
  - VM
    - One vCPU
    - 768 MB of RAM
    - Debian 10
  - Over the network: less than 0.6ms latency and 940 Mbps bandwidth (*iperf*)

Performance (Xen vs KVM)

- Using *Sarracenia*[1] honeypot (VMI-based SSH honeypot)
- System-wide tracing

	KVM		XEN	
	Baseline	With Tracing	Baseline	With Tracing
ls	$0.15s$	$0.37s$	$0.13s$	$0.40s$
wget	$0.57s$	$0.99s$	$0.65s$	$0.98s$

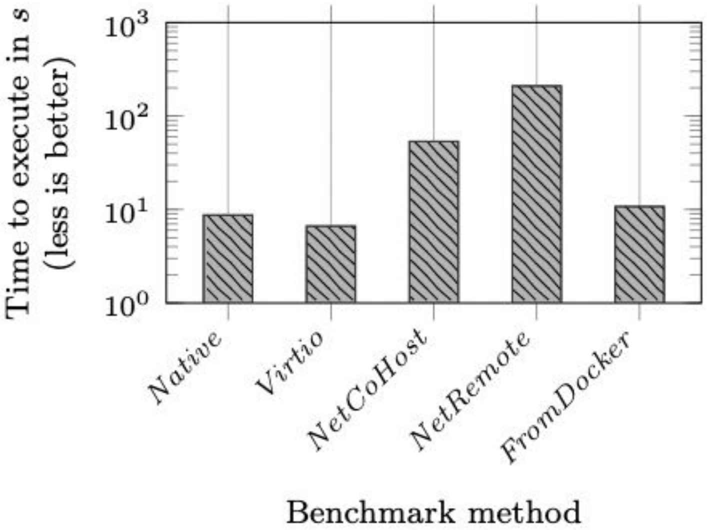
- Function tracing



[1] Sentanoe, S., Taubmann, B. and Reiser, H.P., 2018, November. Sarracenia: enhancing the performance and stealthiness of SSH honeypots using virtual machine introspection. In Nordic Conference on Secure IT Systems (pp. 255-271). Springer, Cham.

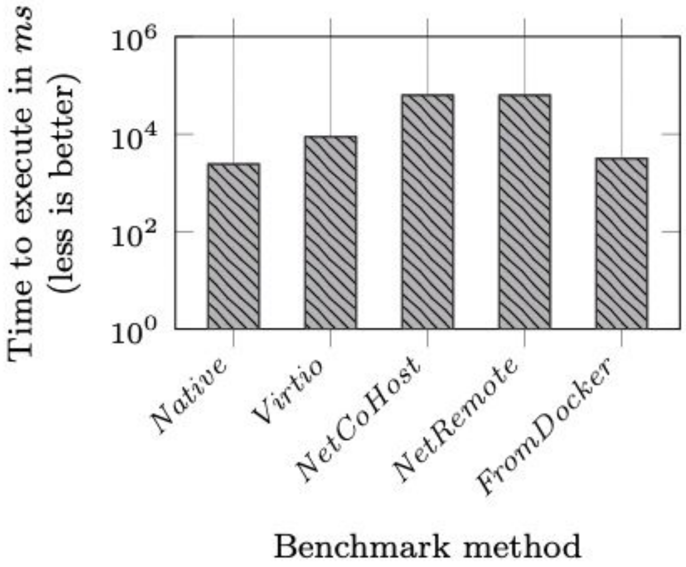
Performance

- Volatility 3 - *lsmod* module
- Created new *DataLayer* so it uses VMI interface
- Runs 50 times



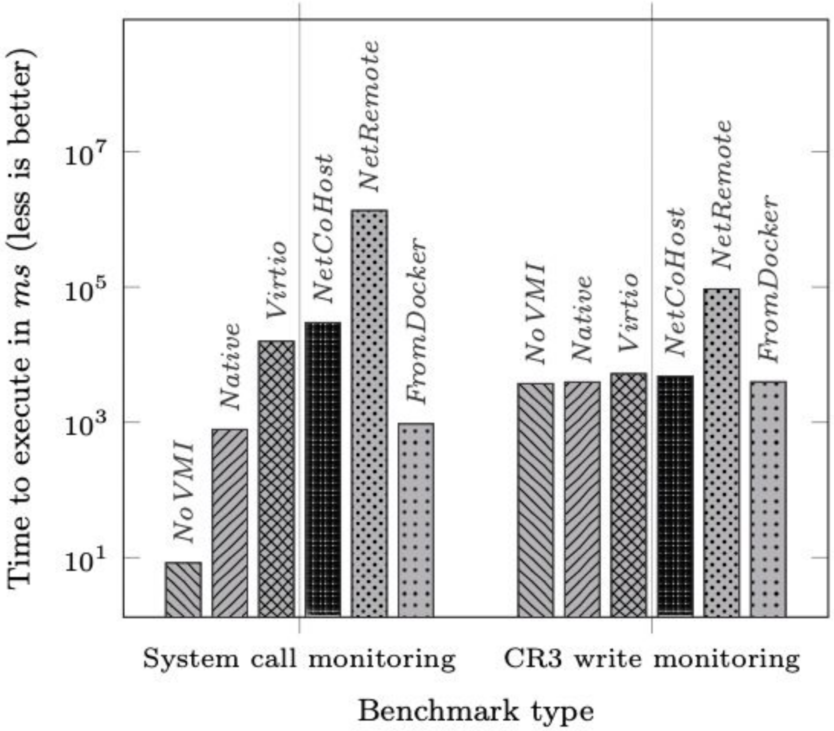
Performance

- Passive VMI
- LibVMI – process list extraction
- Flush cache everytime



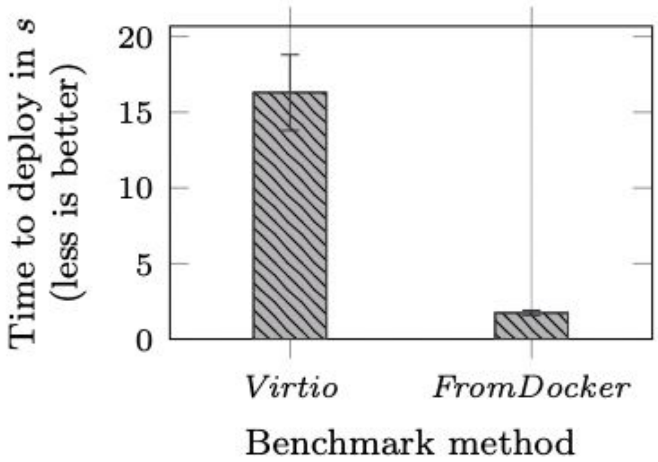
Performance Degradation

- Active VMI – system call handler of *getpid* and *CR3* register



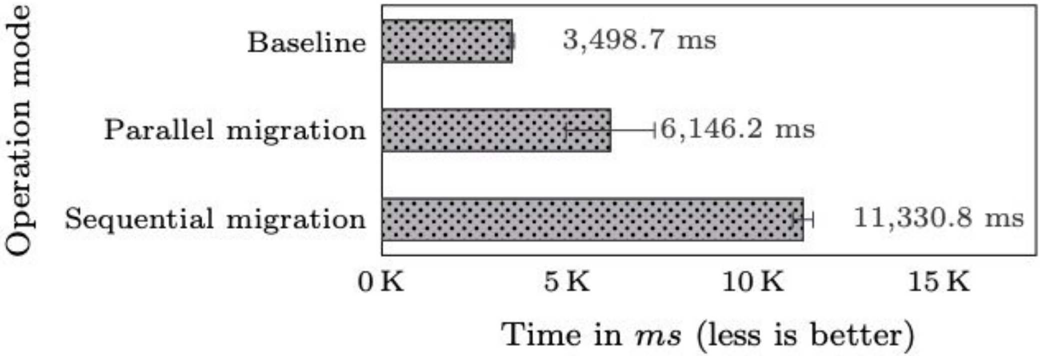
Performance

- *Virtio* and *FromDocker* so far proven to be the best two solutions
- Deployment time of both are acceptable



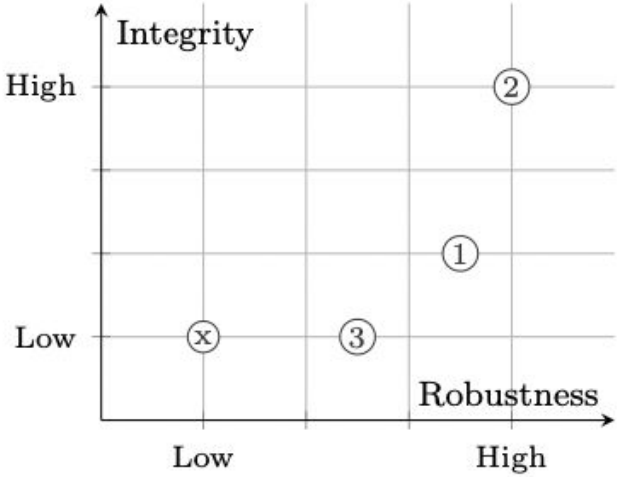
Performance

- VM live migration
- Legend:
  - Baseline ➔ migrate a single VM without any monitor
  - Sequential ➔ migrate monitor VM then target VM
  - Parallel ➔ both at the same time



Robustness & Integrity

- ⓧ → native VMI → neither isolated from the host nor protect integrity
- ① → VM-to-VM → exposes minimum interface
- ② → Over the network → remote access
- ③ → Container → exposes system call interface



Conclusions & Future Work

Conclusions & Future Work

- We introduce *KVMIveggur*
- Flexible and self-service VMI
- Support passive and active
- Integrated to OpenNebula
- Ported several VMI-based applications and Volatility 3

Thank you!

Questions?

Stewart Sentanoe
se@sec.uni-passau.de
University of Passau, Germany