Reducing the Complexity of Contemporary Cloud Computing Systems

Aakash Arora, Chaitra Niddodi, Sibin Mohan

ENGINEERING AT ILLINOIS
INFORMATIONTRUST INSTITUTE

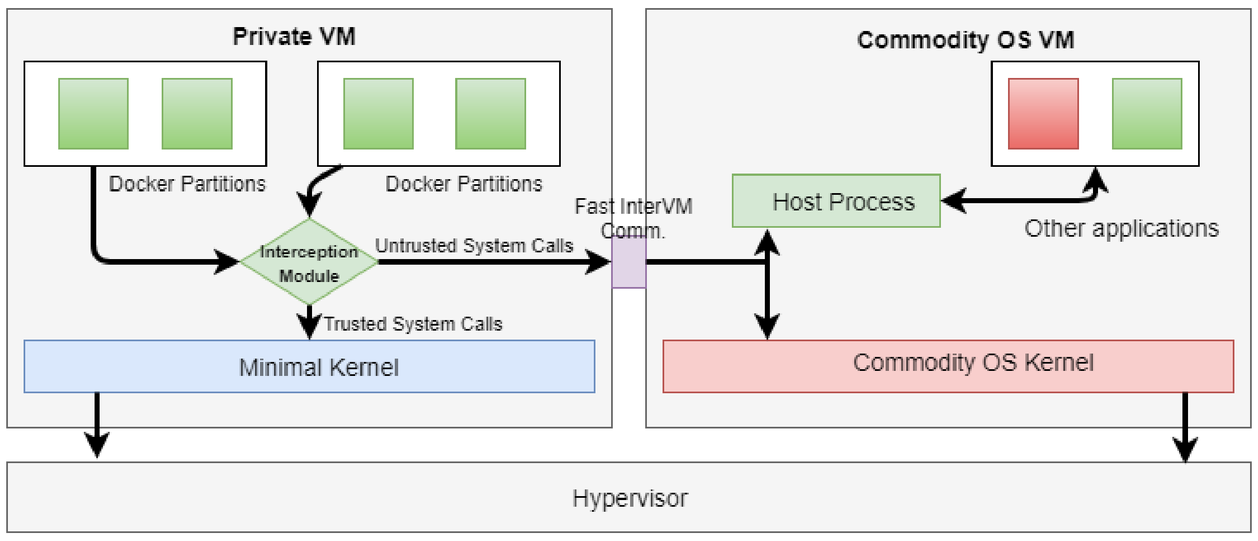
INTRODUCTION

- Containers are easy to use and fast to start compared to virtualization.
- Lack of security in dockers:
  - If a user or application has superuser privileges within the container, the underlying OS can be compromised.
- We present a novel architecture which makes best use of both technologies.
  - Improved isolation of virtualization & robustness of containers.
- Kernel plays a major role in security of any system.
  - Our work involves automating kernel hardening
- Our models supports multiple instances of containers simultaneously while maintaining the real time security aspects.

CHALLENGES

- Reducing the kernel size to absolute minimum – for decreasing the attack surface.
- Establishing of fast, transparent and reliable interVM communication without compromising the isolation.
- Building a light-weight loadable kernel module for intercepting the system calls and routing them to private and commodity OSs in real-time.

SYSTEM ARCHITECTURE



Interception module guarantees that private OS kernel remains corruption free

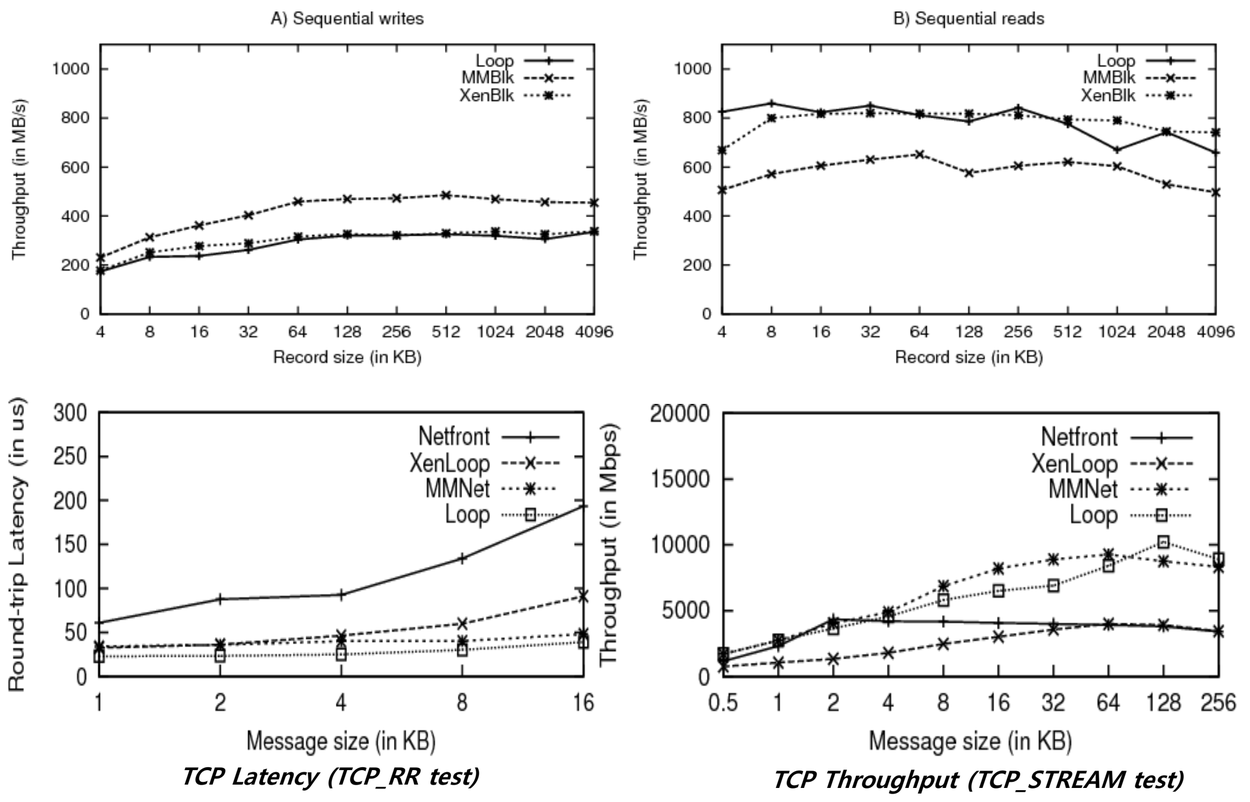
- Our model makes use of output from Cimplifier tool.
  - It divides a docker application into simpler-containers.
  - Ensures privilege separation.
- Private OS kernel is reduced to bare minimum.
  - Based on the requirements of the docker applications.
- Unwanted sys calls were removed from sys call table to further increase security.
- Interception module transparently routes each system call.
  - Trusted ones to Private OS Kernel otherwise to comm.OS kernel.
- Each docker can have different set of trusted and untrusted system calls.
- Interception module should also translate the file descriptors used by two VMs, as both are independent of each other.

RULE OF THUMB

Docker partitions (output from Cimplifier tool) can be placed in private or commodity VM based on its purpose :

- User Interface – Commodity OS VM
- Network – Commodity OS VM
- Persistent Storage (file objects) – Commodity OS VM
- Randomness and system time - Private OS VM
- Memory - Private OS VM
- Any combination of above (with private OS)- Private OS VM

SURVEY OF VARIOUS INTER-VM COMMUNICATION APIs



TCP Latency (TCP_RR test) ***TCP Throughput (TCP_STREAM test)***

- For all the various tests FIDO (MMNet and MMBlk) out-perform others in general, but it relies on relaxed trust model which is unfit for our application.
- XenLoop (network based) ensures higher degree of security and comparable results for smaller message sizes.

CONCLUSION AND FUTURE WORK

❑ *Our model ensures* :

✓ Underlying private OS kernel remains corruption free and ensures isolation among various docker containers.

✓ Containers in private VM remains unaffected by applications in commodity OS VM.

✓ Even if trusted docker is compromised other instances of dockers will remain unaffected.

❑ *This initial effort can be extended in several directions:*

- Testing the prototype against real-world threats.
- Using non-serialized interVM communication.
- Enforcing resource constrains based on specific user privileges across VMs.

ACKNOWLEDGEMENT

- Dr. Sibin Mohan
- Prof. Somesh Jha
- Prof. David Lie
- Dr. Vaibhav Rastogi