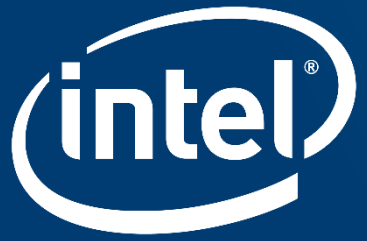


Alibaba Cloud | Worldwide Cloud Services Partner

SECURED CONTAINERS WITH SGX AND GOLANG SUPPORT

Helping protect secrets inside cloud native applications

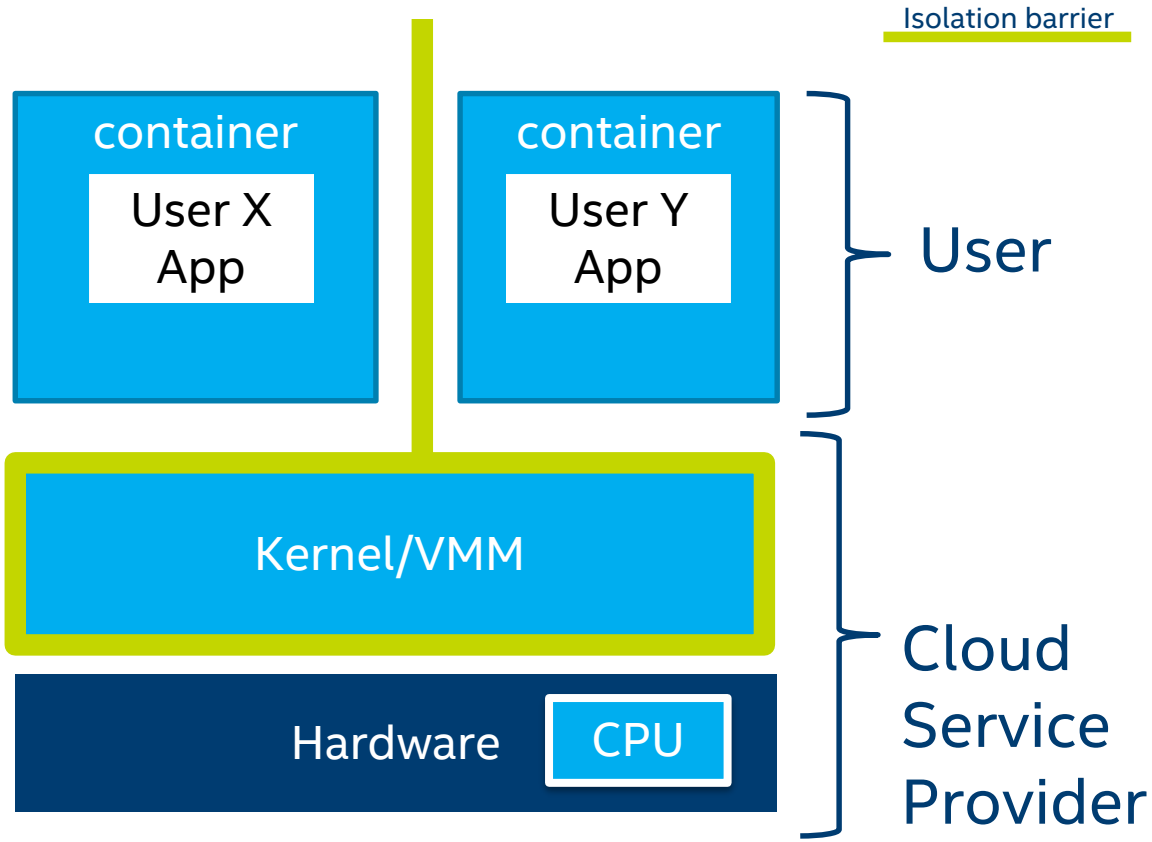
Isaku Yamahata isaku.yamahata@intel.com

Xiaoning Li xiaoning.li@alibaba-inc.com

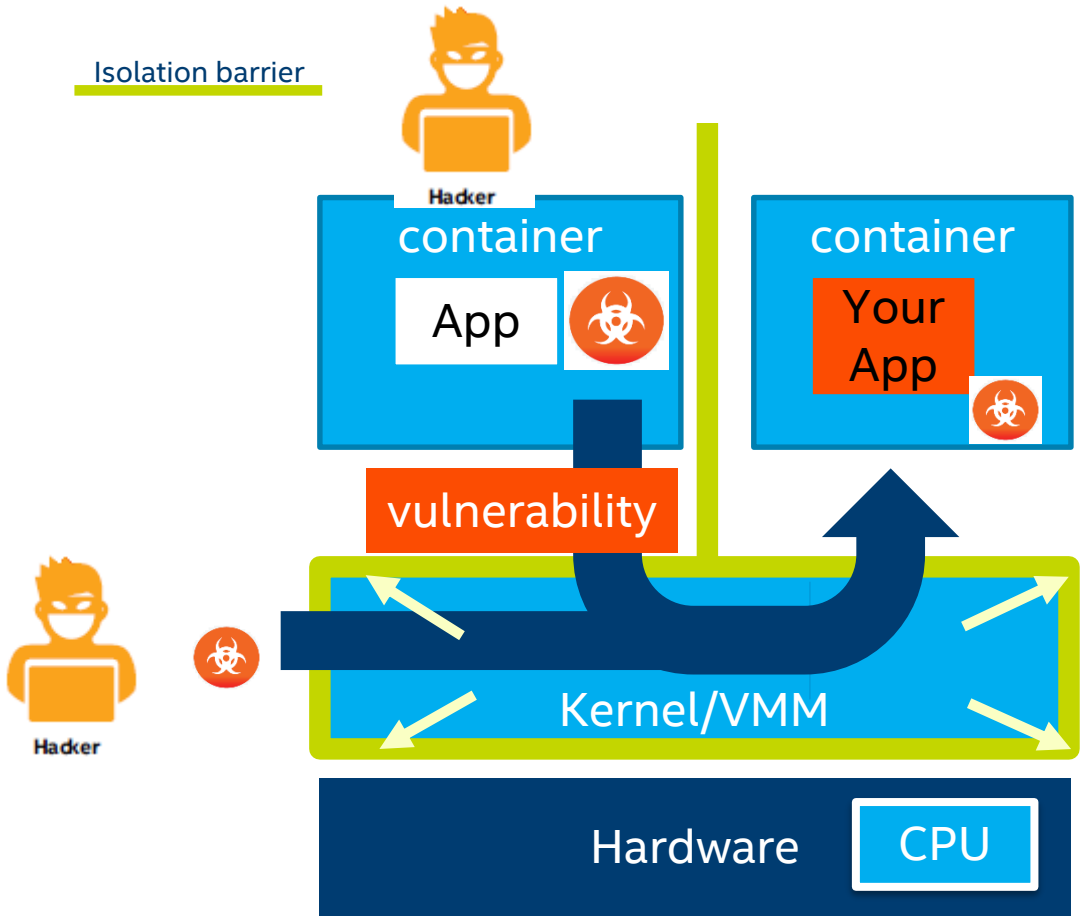
Kubecon China 2019
June 26, 2019

Secured containers in the cloud environment

- In a public cloud environment, the platform is not controlled by users.
- Users manage only their applications.
- Kernel/VMM are managed by the cloud provider



Traditional threat model

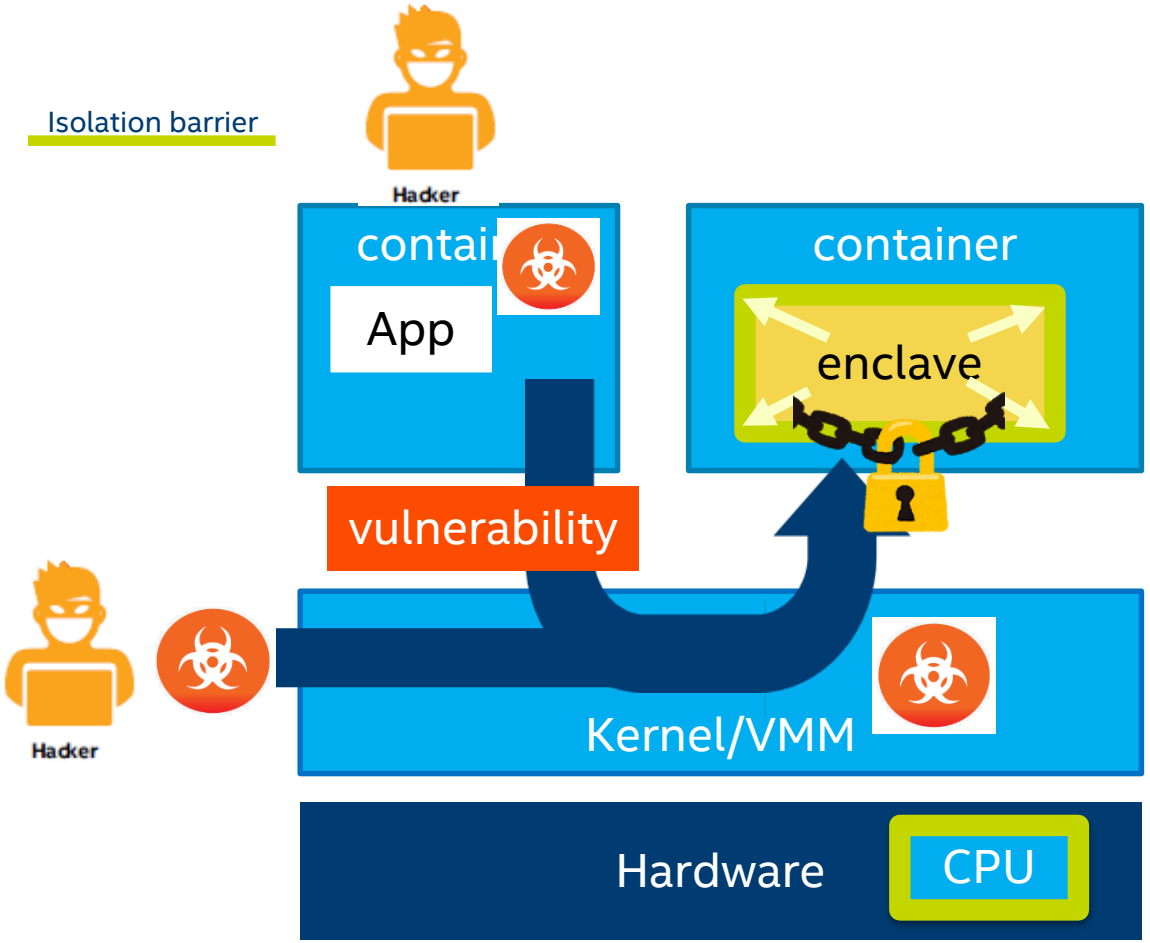


- Cloud users trust Kernel/VMM
- Protect Kernel/VMM against attack
- If kernel/VMM is vulnerable, whole system may become vulnerable.

=>

- Your secret needs to be protected from host OS/VMM, and platform firmware
- TEE: Trusted Execution Environment

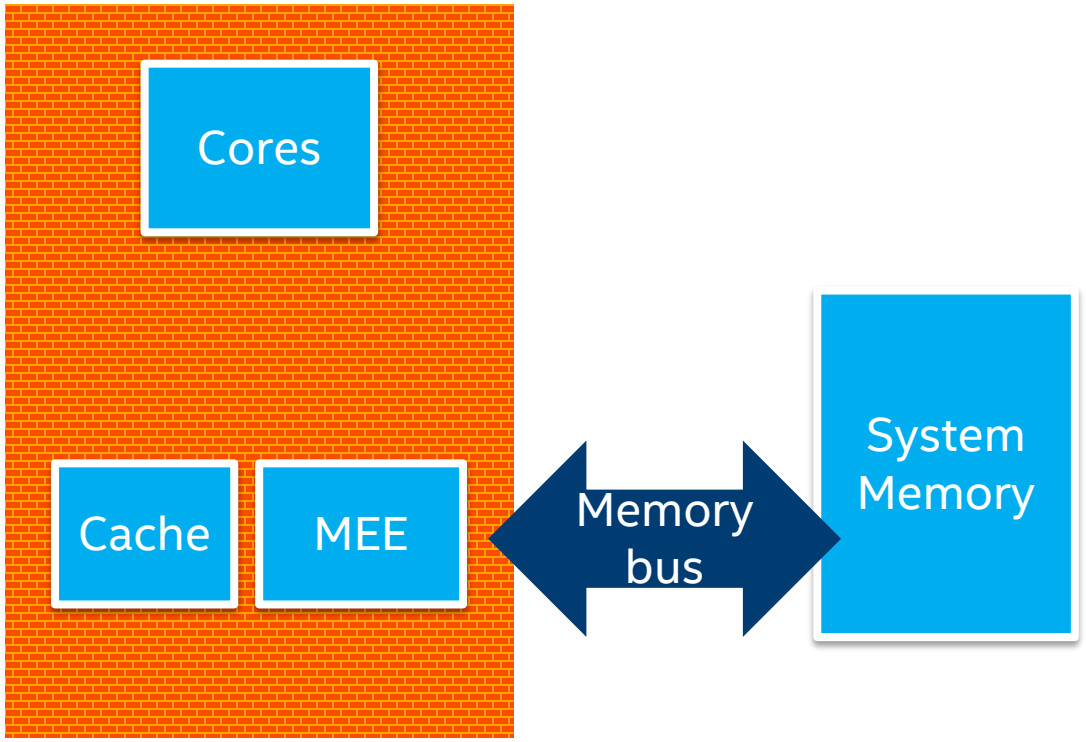
Trusted Execution Environment(TEE)



- Cloud users work with enclaves with security features.
- Cloud users doesn't trust Kernel.
- Even if kernel is vulnerable, your code/data should have more security protection.

What's SGX?

- Software Guard Extension(SGX) as TEE
  - Enclave is trusted.
- Enclave
  - Special encrypted memory region
    - DRAM/bus are encrypted
  - Boot with "measurement"
  - Restricted execution environment
    - E.g. rdtsc/syscall instructions are prohibited. #UD



CPU Package

MEE:Memory Encryption Engine

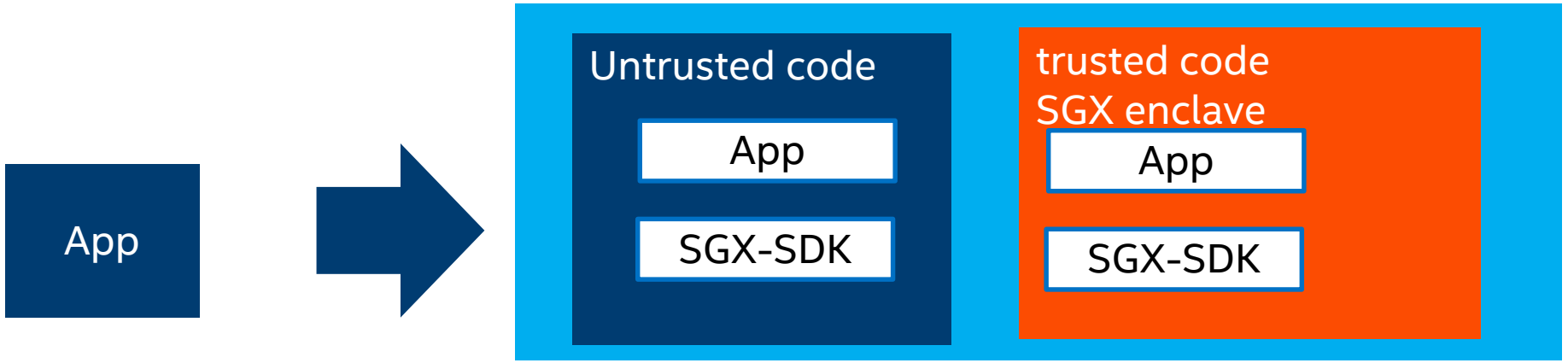
Adapting SGX

- Typically SGX requires application modification at source code level
  - Split protected code+data, and generic/unprotected code
  - Put protected code into SGX enclave
- Major reorganization of application

Refactoring the application at source code level

Mitigating costs to adopt SGX

- As mitigation, the SGX SDK library is provided.
- It helps reduce migration cost.



But still some refactoring at source code level is still needed

Comparison of adapting SGX

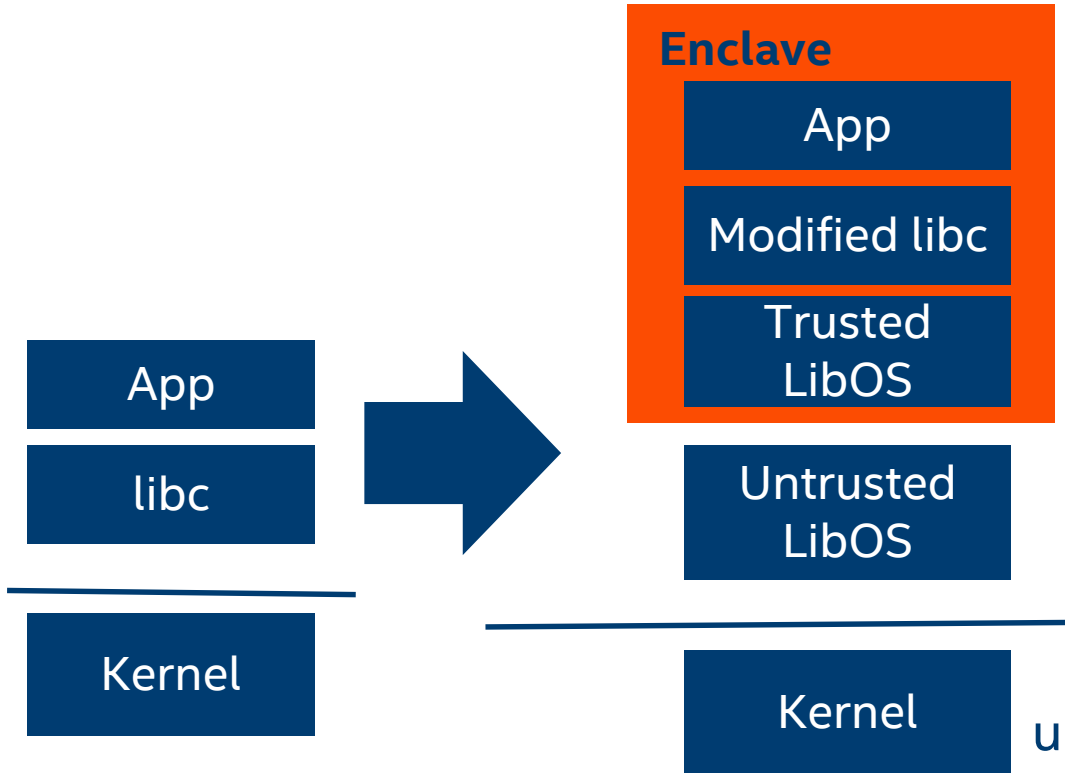
Target

item	?	Use SGX-SDK library	Adapt SGX natively
description	?	Modify/refactor application at source code level	Modify/refactor application at source code level
Adapting Cost	Low	Mid	High
TCB size	?	Mid (a part of app + sgx-sdk)	(can be) Small (a part of app)

Note: which method is better depends on your use case and available resource

Library OS for secured container

- Bring your own binary.
- Library OS supports unmodified user binary to run within enclave



- LibOS handles system calls
- Various design choices
  - Hooking system call
    - Modified shared Library(glibc)
      - Syscall => function call
    - Trap and emulate
  - Syscall emulation
    - Typicall Linux system call
  - Syscall shielding

What about Golang?

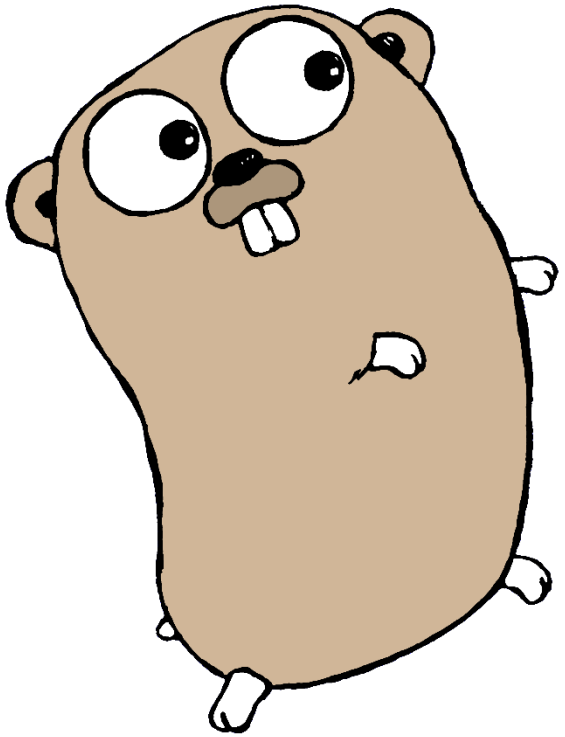
- Golang is one of the more popular languages for cloud native applications

PULL REQUESTS

Year: 2019 | Quarter: 2

# Ranking	Programming Language	Percentage (Change)	Trend
1	JavaScript	19.922% (-2.425%)	
2	Python	17.803% (+1.519%)	
3	Java	10.482% (+0.591%)	
4	Go	7.916% (+0.295%)	
5	C++	7.253% (+0.167%)	
6	Ruby	6.296% (-0.249%)	
7	PHP	5.515% (-0.285%)	
8	TypeScript	5.415% (+0.641%)	
9	C#	4.001% (+0.659%)	
10	C	3.190% (+0.248%)	

https://madnight.github.io/githut/#/pull_requests/2019/2



https://go.googlesource.com/go/+/refs/heads/master/doc/gopher/gophercolor.png

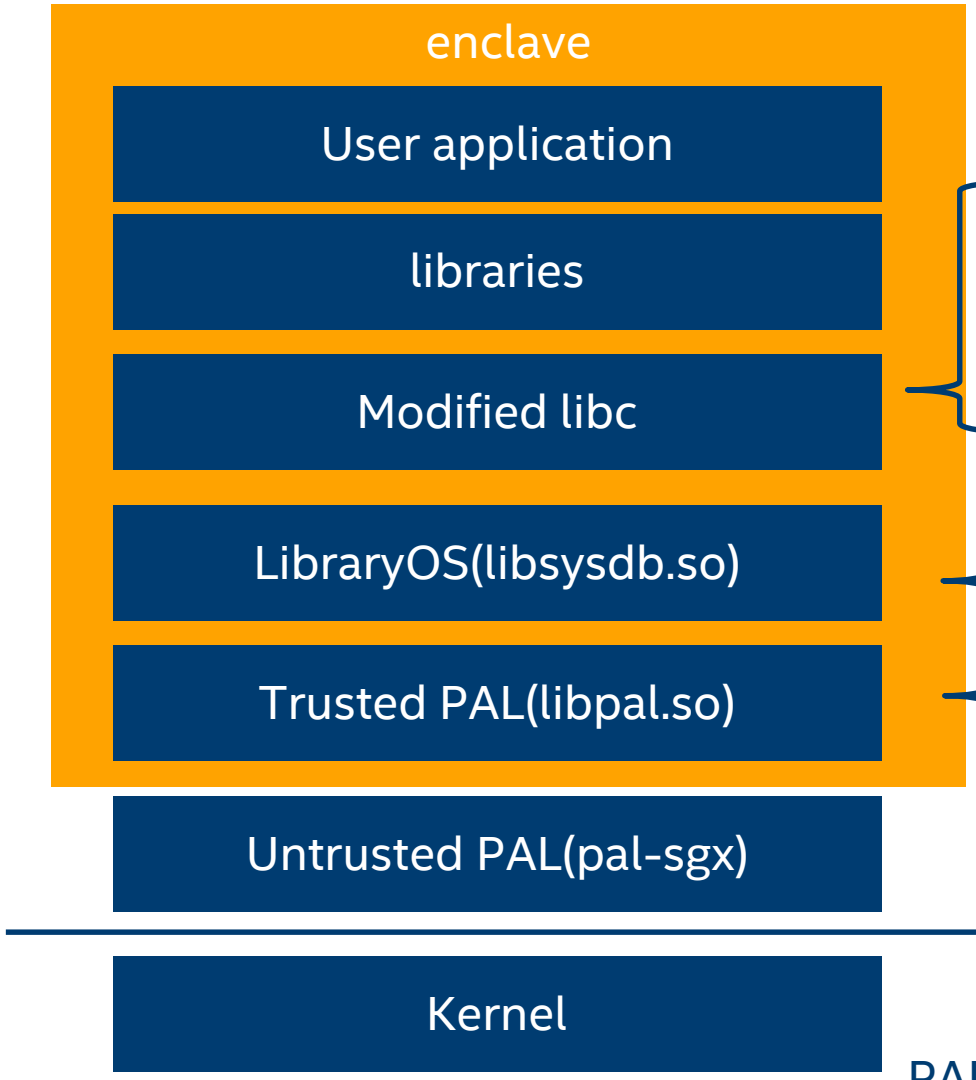
Challenges of LibOS for Golang

- It behaves differently from C/C++ programs
  - LibOS compatibility
- Go runtime
  - Golang runtime doesn't use libc. It has its own runtime written in Go
- Application binaries are statically linked
- Goroutine: Small stack size (128kB)
- Need for accurate emulation of POSIX signals
- Syscall emulation quality
  - Golang uses system calls differently from glibc

Graphene LibOS: SGX supported

- https://grapheneproject.io/
- A Library OS for Unmodified Applications
- Linux System call emulation
- Multi-process support
- Multi-platform support

Graphene-SGX architecture



Hooking system call
function call instead of syscall instruction

Linux system call emulation
with limited number of host system call

System call shielding against Iago attack
Check if system call result is sane.

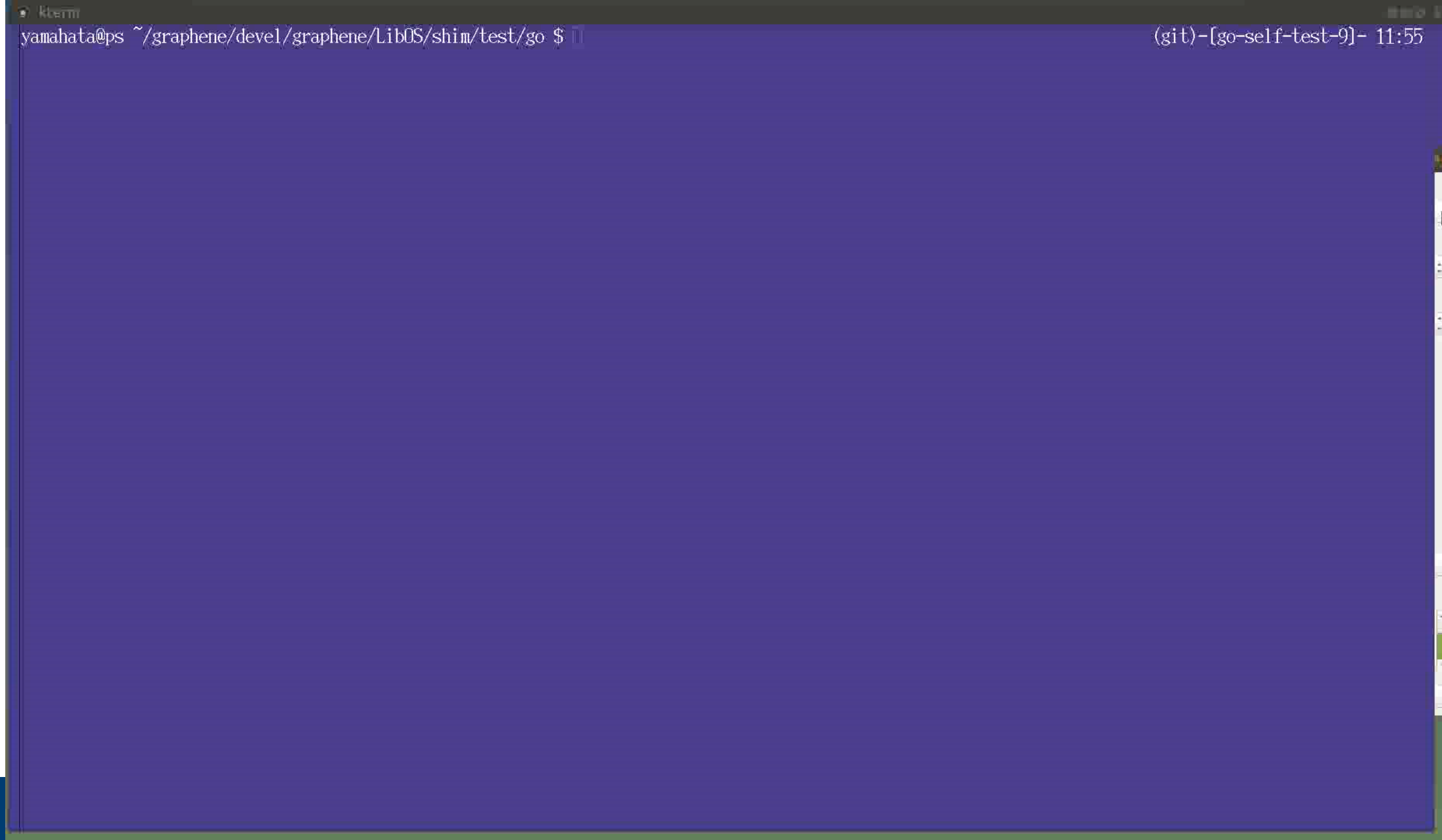
Limited number of system call is used

PAL: Platform Adaptation Layer

Graphene-SGX for Golang support

- Go runtime
  - Syscall emulation and optimization
  - vDSO
- Dedicated stack
- Accurate emulation of POSIX signals
- Improved syscall emulation quality
  - Golang uses system call differently from glibc

Demo


kterm
yamahata@ps ~/graphene/devel/graphene/LibOS/shim/test/go $
(git)-[go-self-test-9]- 11:55

Summary

- Trusted Execution Environment(TEE) is important in cloud environment era
- SGX is a basic building block for TEE
- Library OS is one of the solutions to help enable secured container at reduced cost
- Graphene-SGX is open-source solution and its Golang support is coming

Thank you

Call to Action

- Give it a try
- Contribute to Graphene project

Resources

- https://software.intel.com/en-us/sgx
- https://grapheneproject.io/
- https://github.com/oscarlab/graphene
- https://golang.org/

Legal Disclaimer

Intel provides these materials as-is, with no express or implied warranties.

All products, dates, and figures specified are preliminary, based on current expectations, and are subject to change without notice.

Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

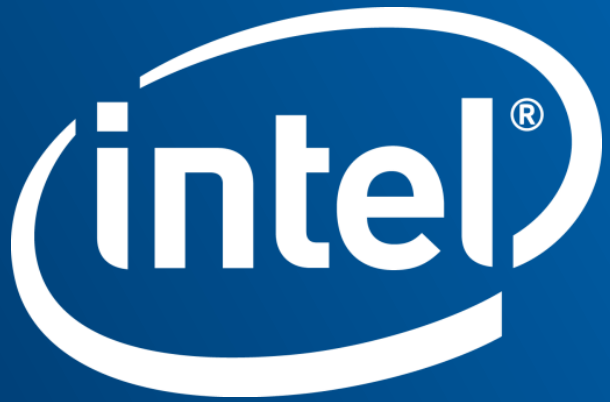
Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://intel.com.

Some results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others.

© Intel Corporation


intel