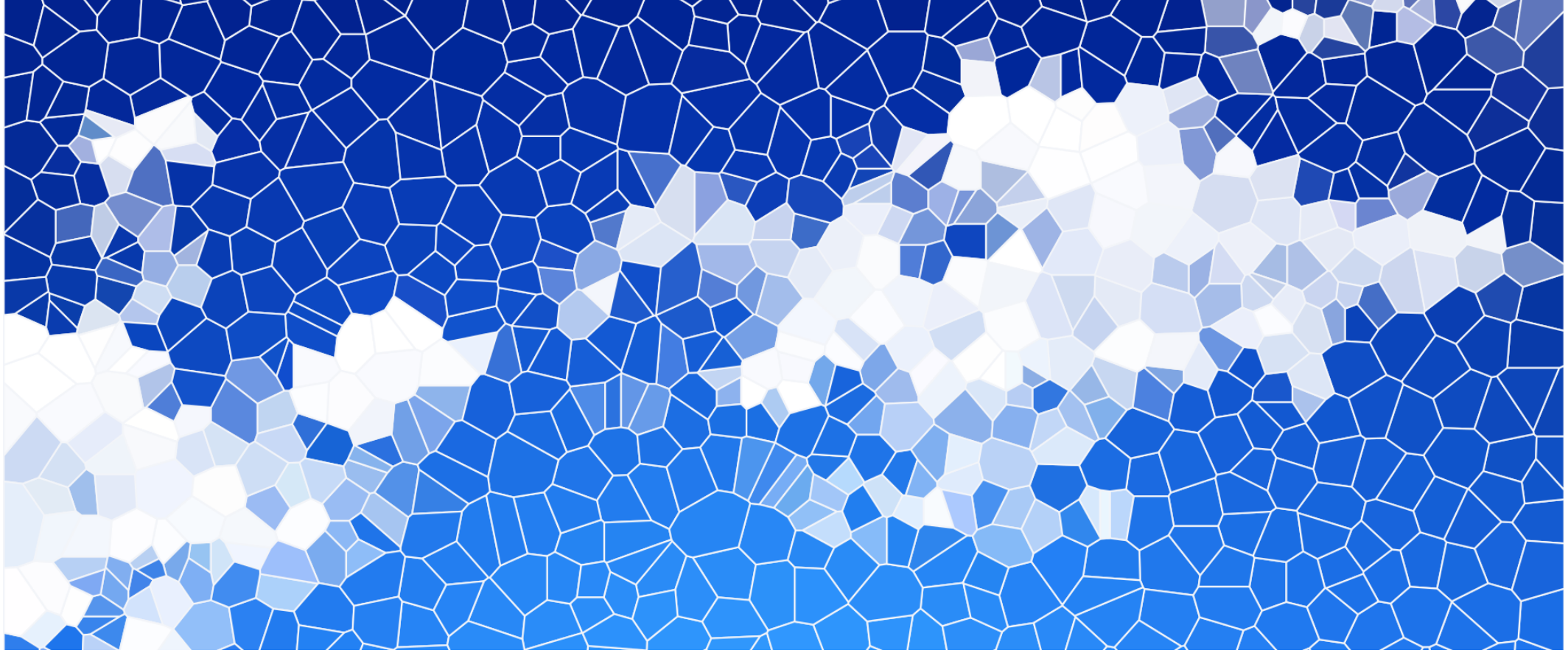
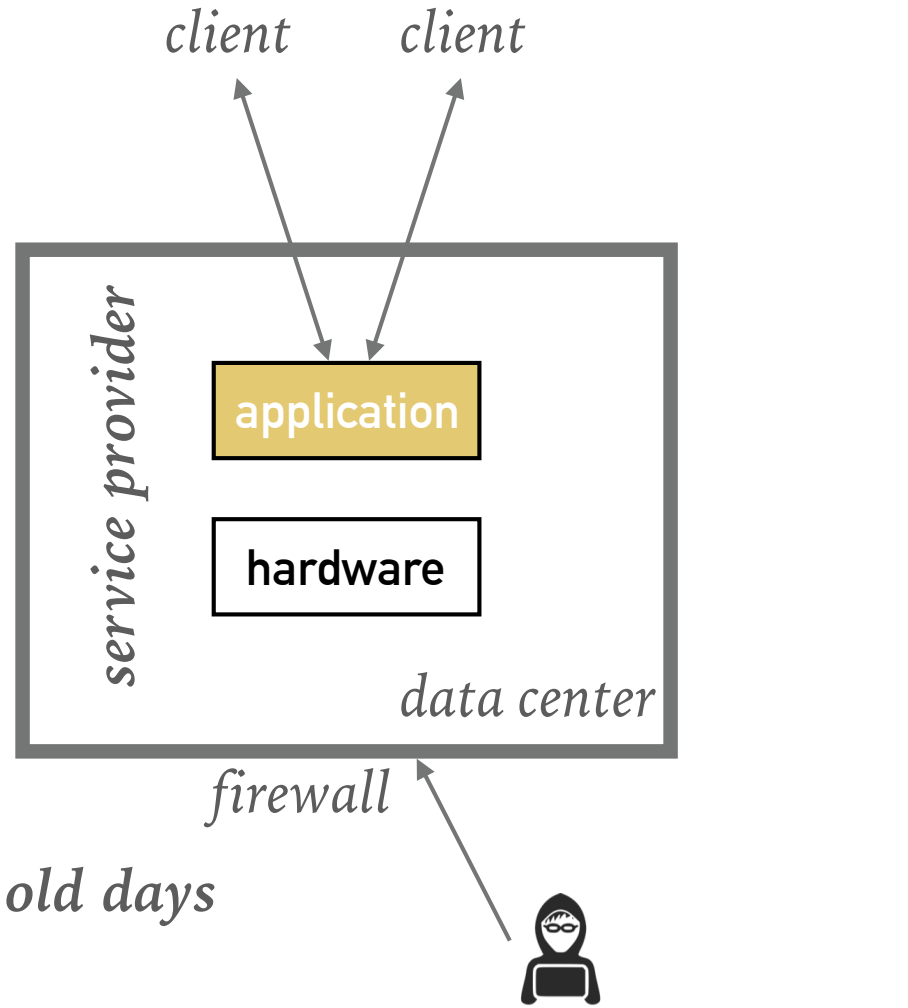


SCONE

Secure CONtainer Environment

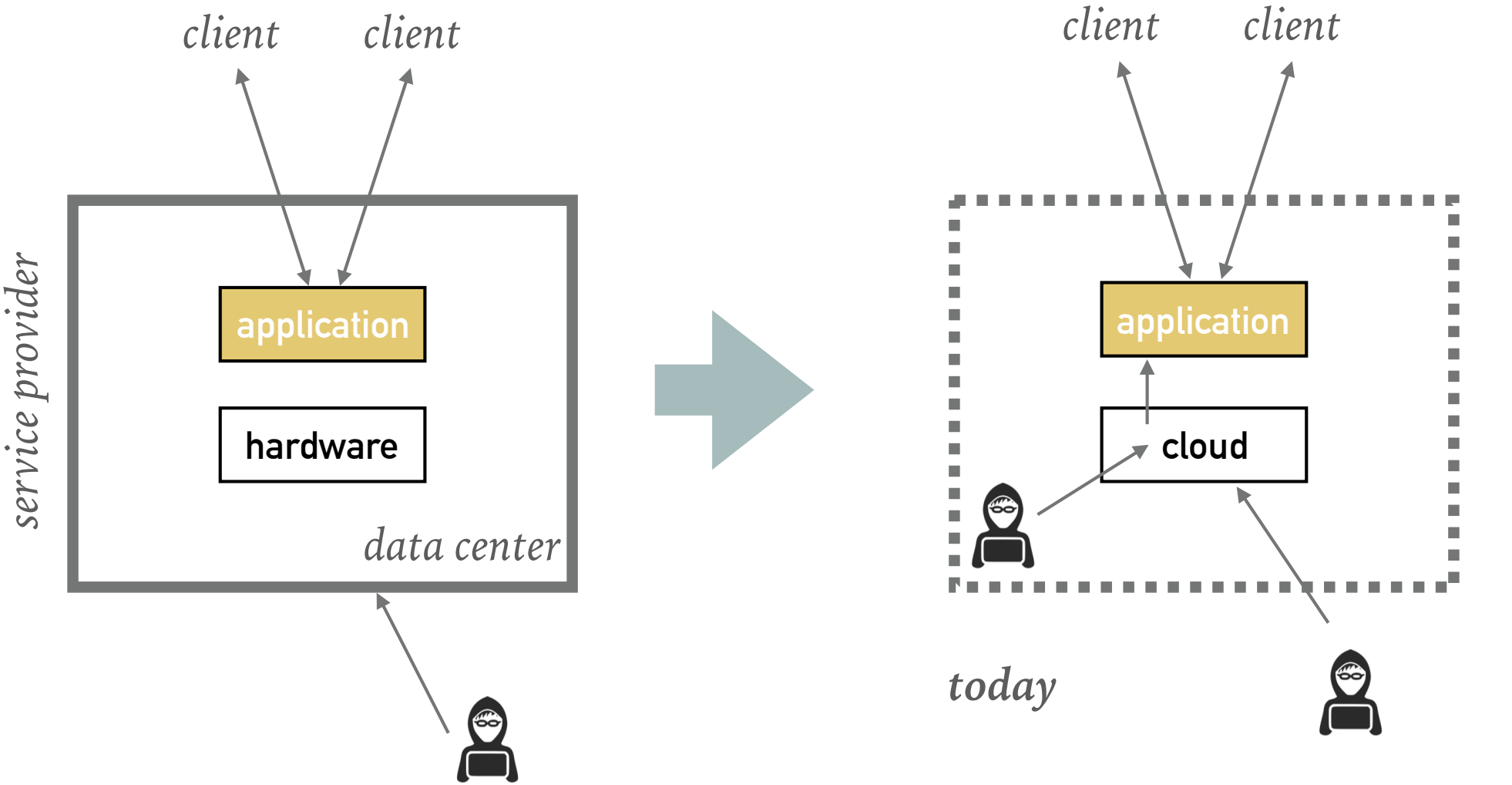
Christof Fetzer, TU Dresden, Germany

MOTIVATION



Prevent unauthorized access

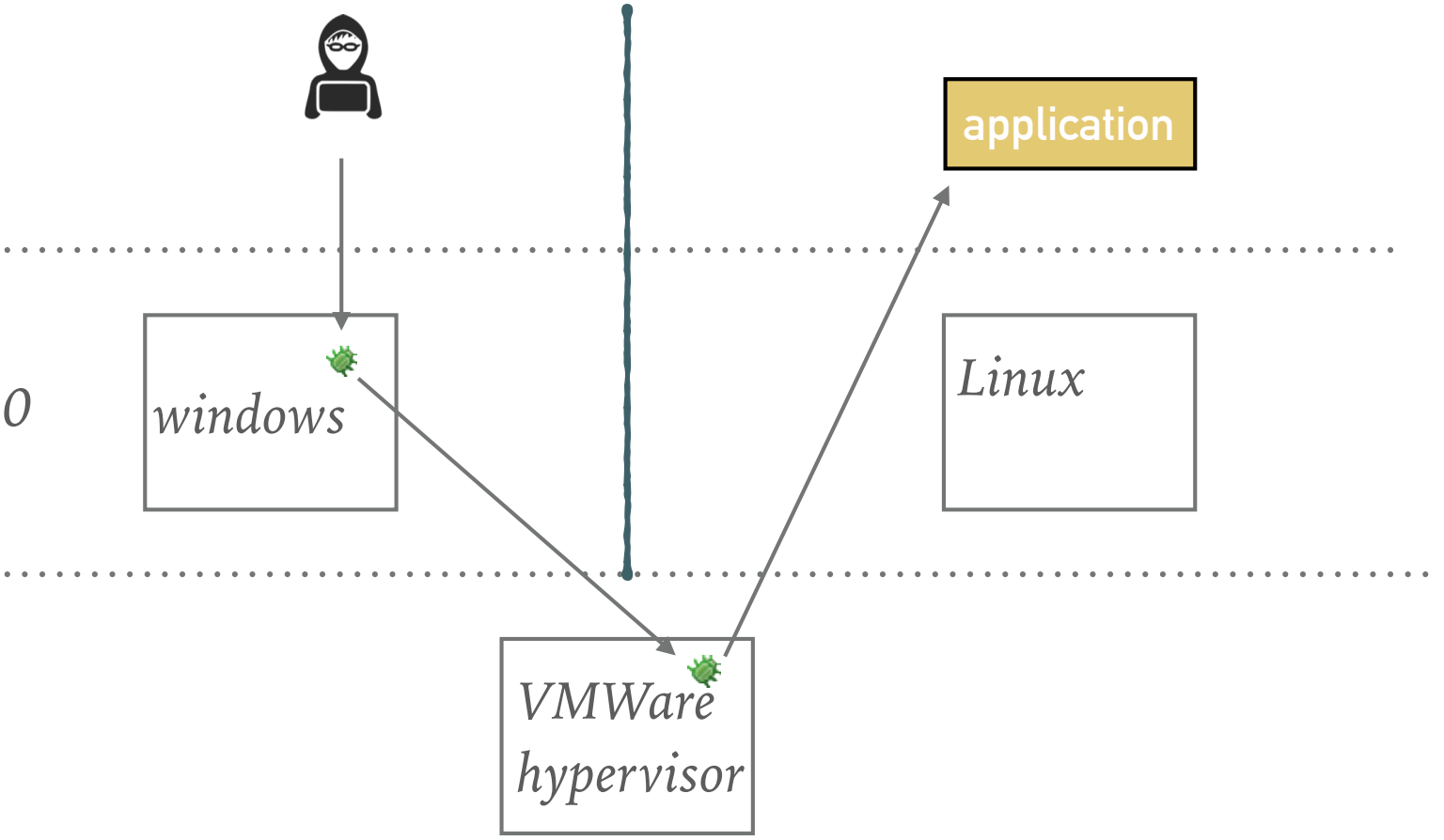
OBJECTIVE: PROTECT CONFIDENTIALITY & INTEGRITY



Prevent unauthorized access

ONLY ACADEMIC ISSUE?

2017 hacking contest



bug in Windows 10

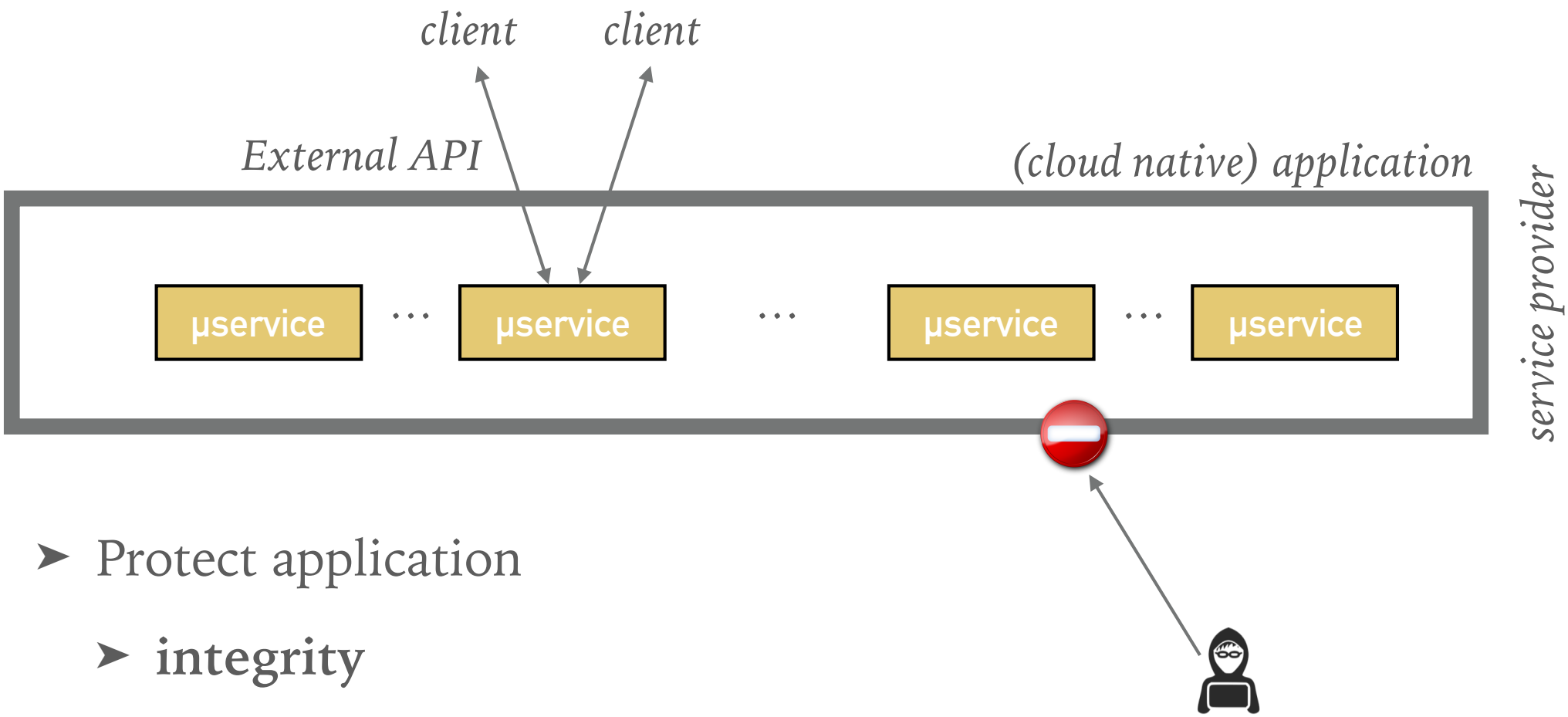
bug in VMware Workstation

https://arstechnica.com/security/2017/03/hack-that-escapes-vm-by-exploiting-edge-browser-fetches-105000-at-pwn2own/

GENERAL APPROACH

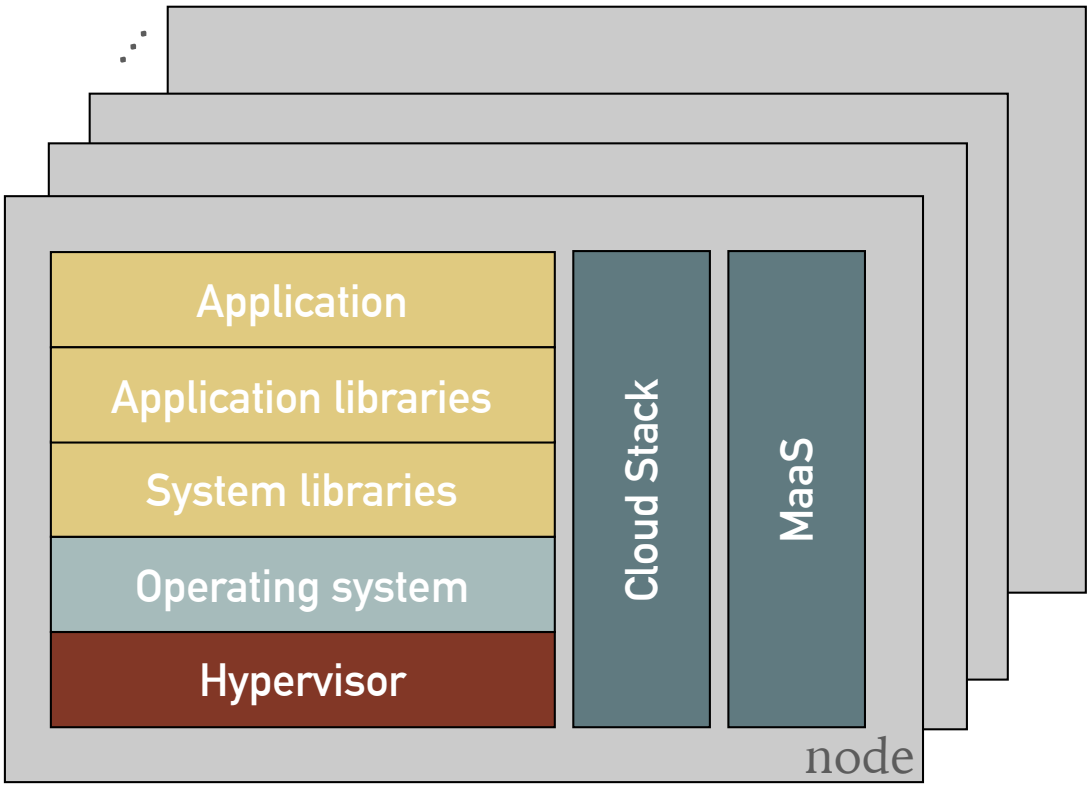
- „Firewall“ around the application
  - **application-oriented security**
- **Cloud-Native Applications**
  - set of microservices

APPROACH: APPLICATION-ORIENTED SECURITY



- Protect application
  - **integrity**
  - **confidentiality**

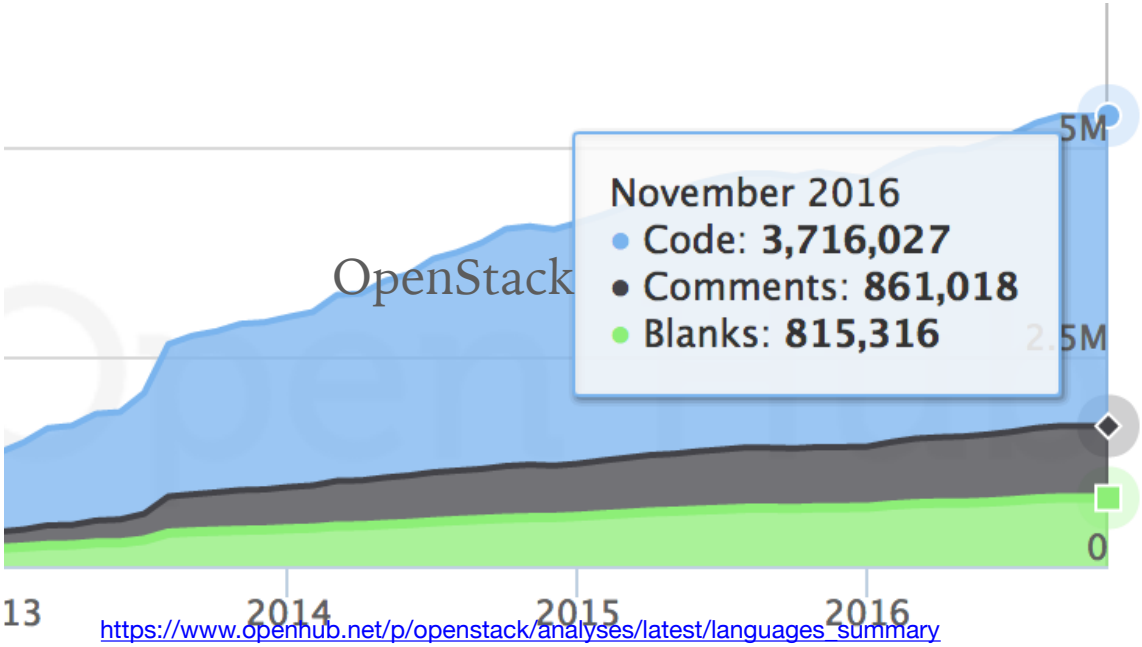
DEFENDER'S DILEMMA



cloud software stack

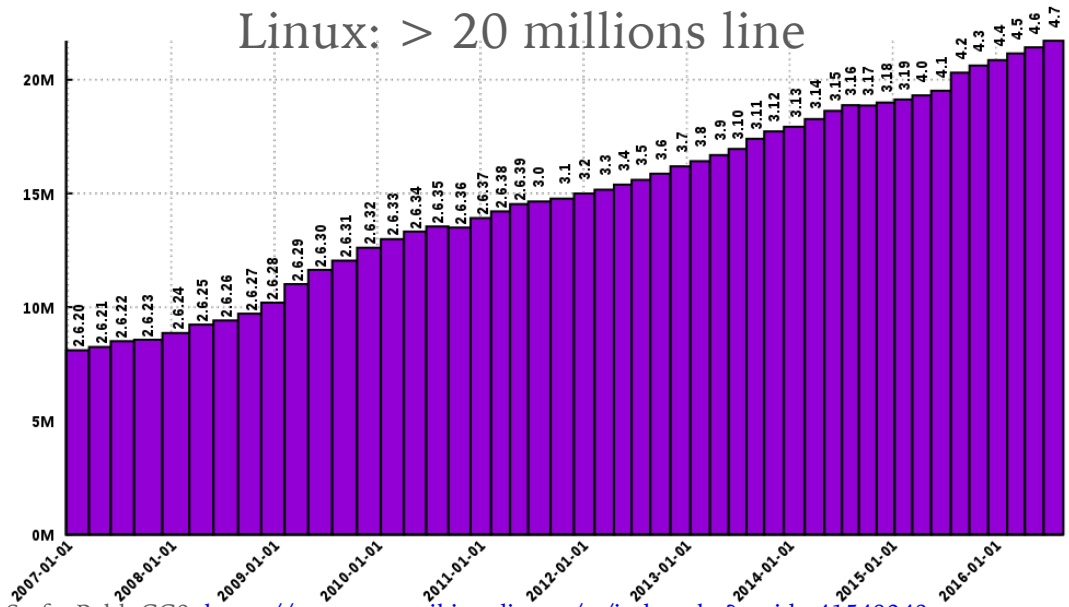
- **Attackers**:
  - success by exploiting a single vulnerability
- Defender:
  - must protect against every vulnerability
    - **not only in application**
  - millions of lines of source code

CLOUD SOFTWARE STACK



https://www.openhub.net/p/openstack/analyses/latest/languages_summary

- Applications run on top of software stack
  - millions of lines of code
- Cloud stack consists of
  - VM/container engine
  - operating system
  - hypervisor
  - node management service



StefanPohl, CC0, https://commons.wikimedia.org/w/index.php?curid=41549243

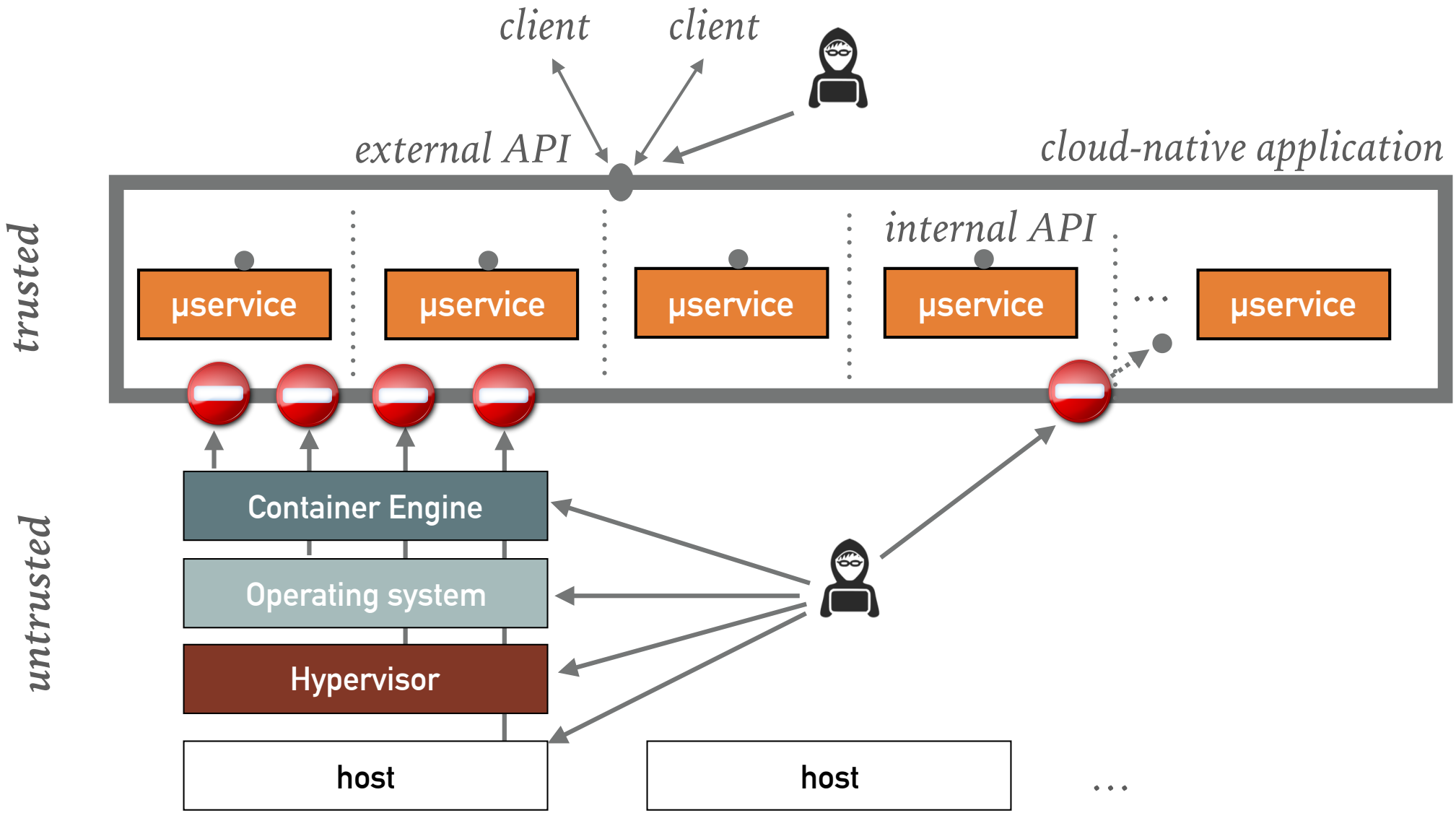
VULNERABILITIES

- **Coverity reports:**
  - 1 defect per 1700 lines of code
- **Kernel self protection project:**
  - 500 security bugs fixed in Linux during the last 5 years
  - each bug stayed about 5 years inside kernel
- **Coverity:**
  - quality of closed source software is not better than open source software

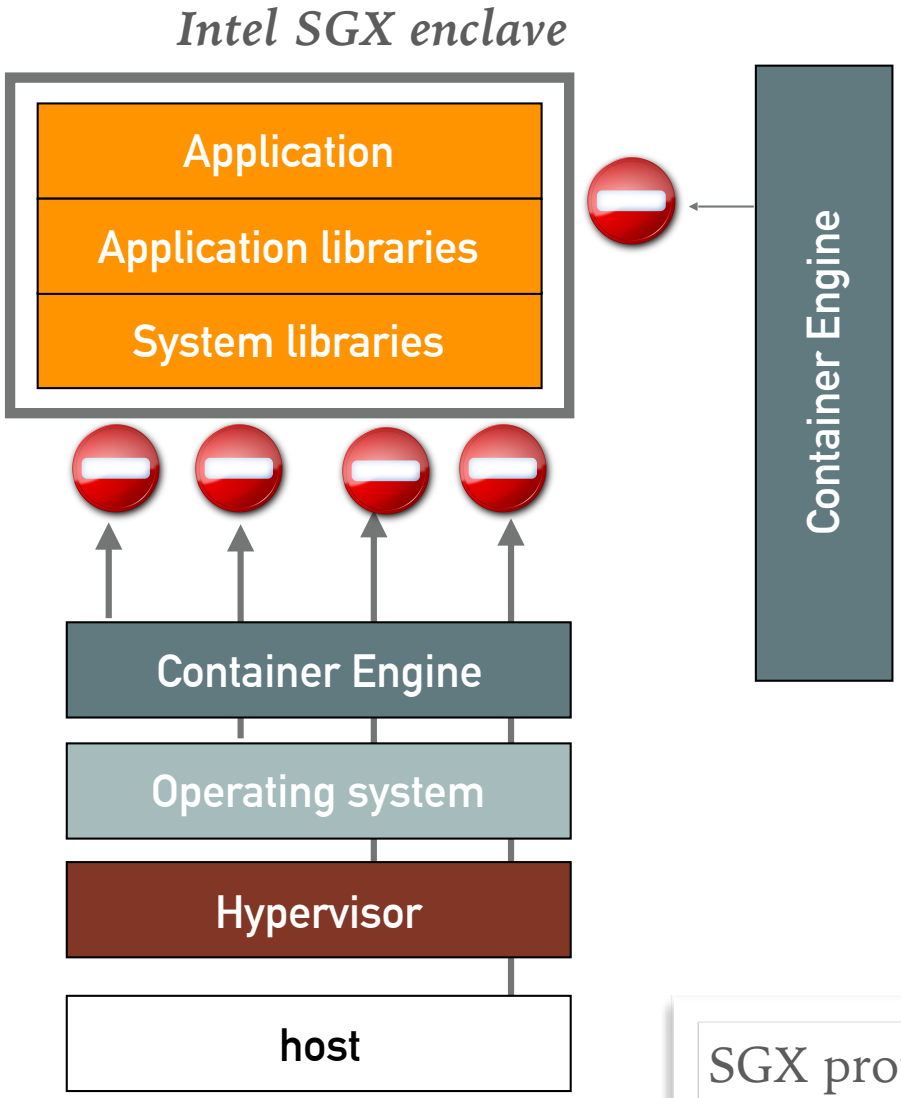
[KSPP] Kees Cook, The State of Kernel Self Protection Project, Linux Security Summit (LSS), 2016

[Coverity] Open Source Report 2014 - Coverity, go.coverity.com/rs/157-LQW.../2014-Coverity-Scan-Report.pdf

APPLICATION-ORIENTED SECURITY



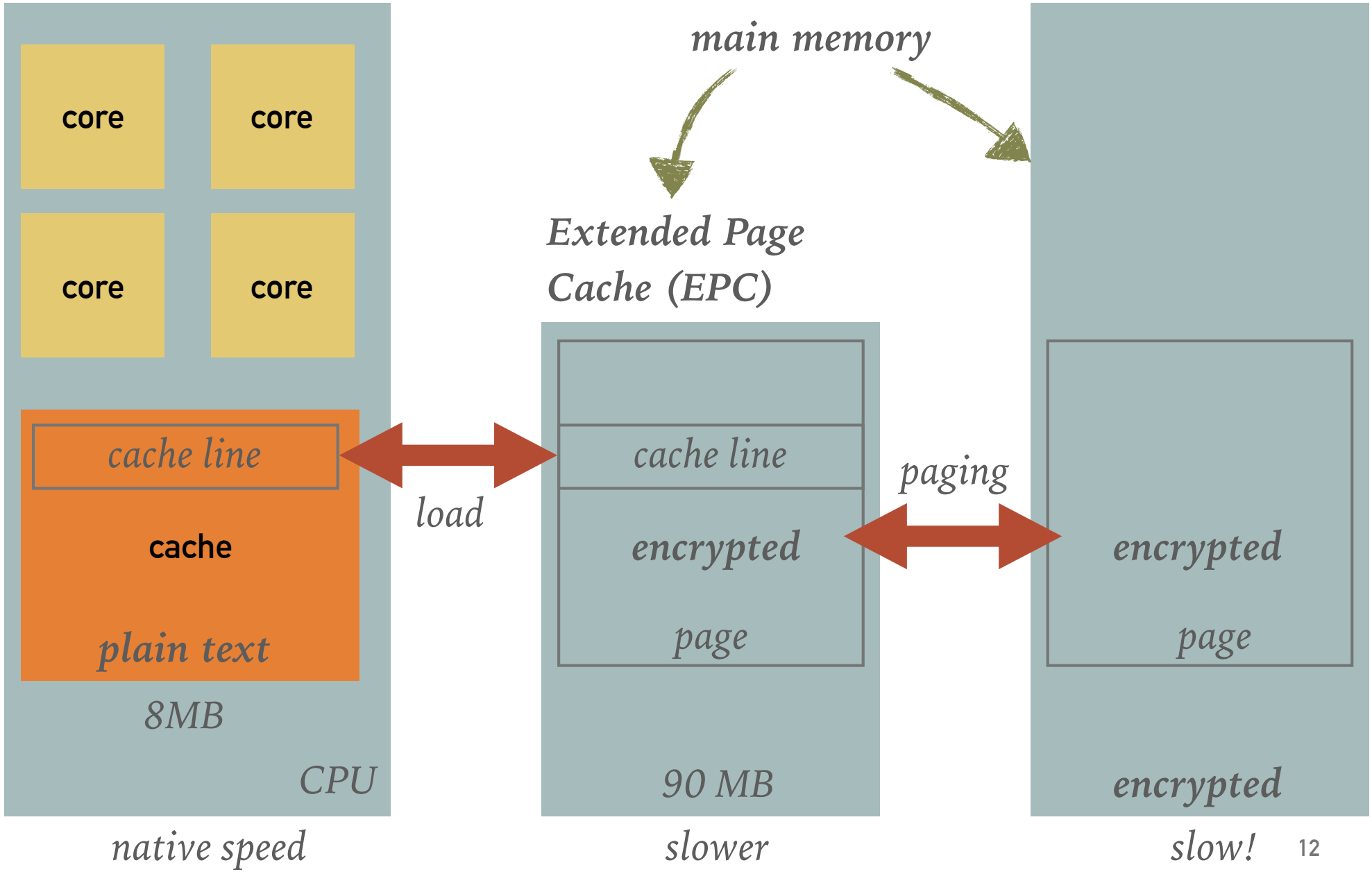
APPLICATION PROTECTION



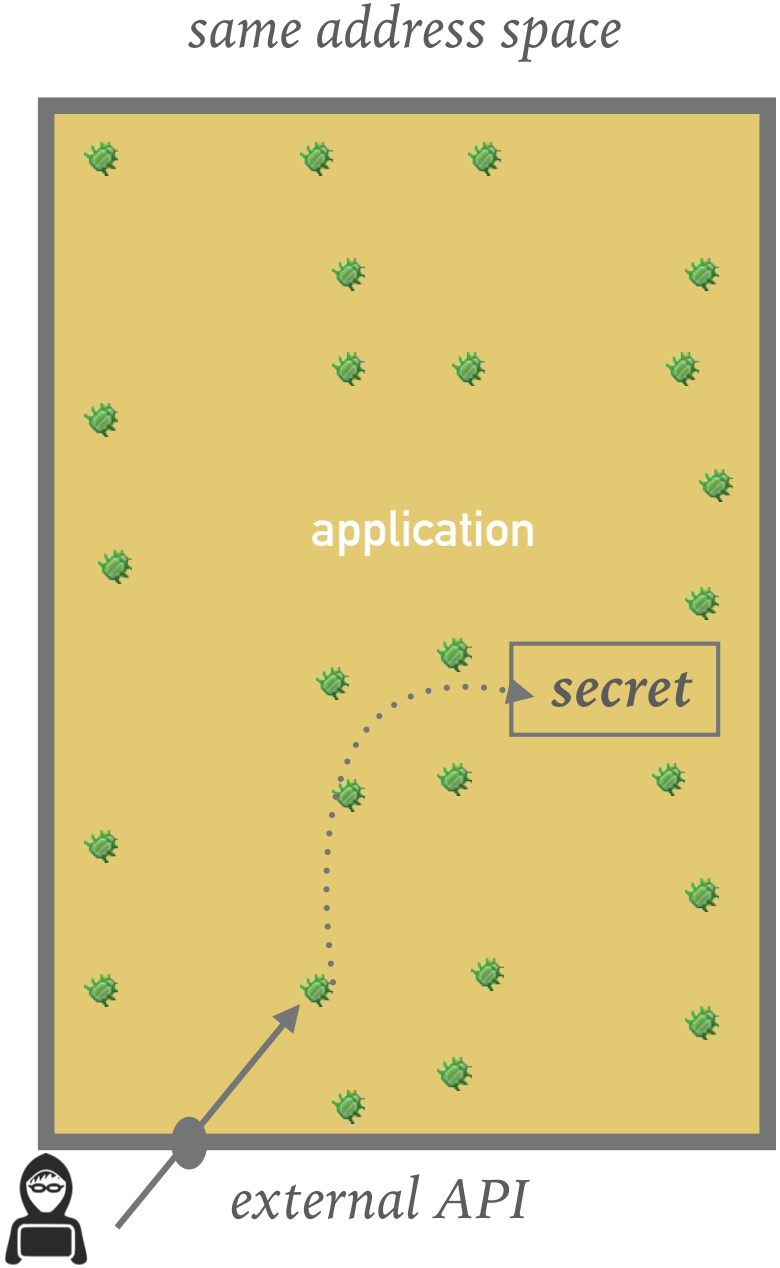
- Intel SGX protects application's
  - confidentiality
  - integrity
- by preventing accesses to
  - application state
  - encrypting main memory

SGX protects application from accesses from outside

SGX PERFORMANCE



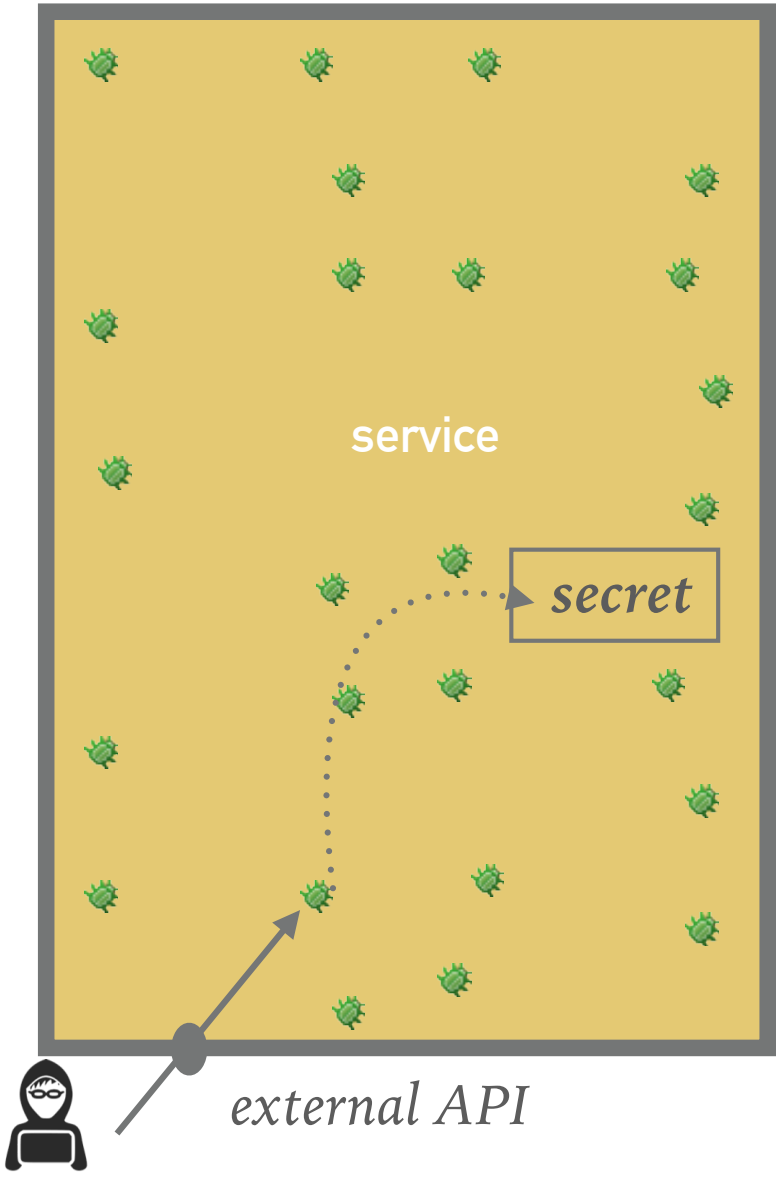
SAME PROBLEM: BUGS!



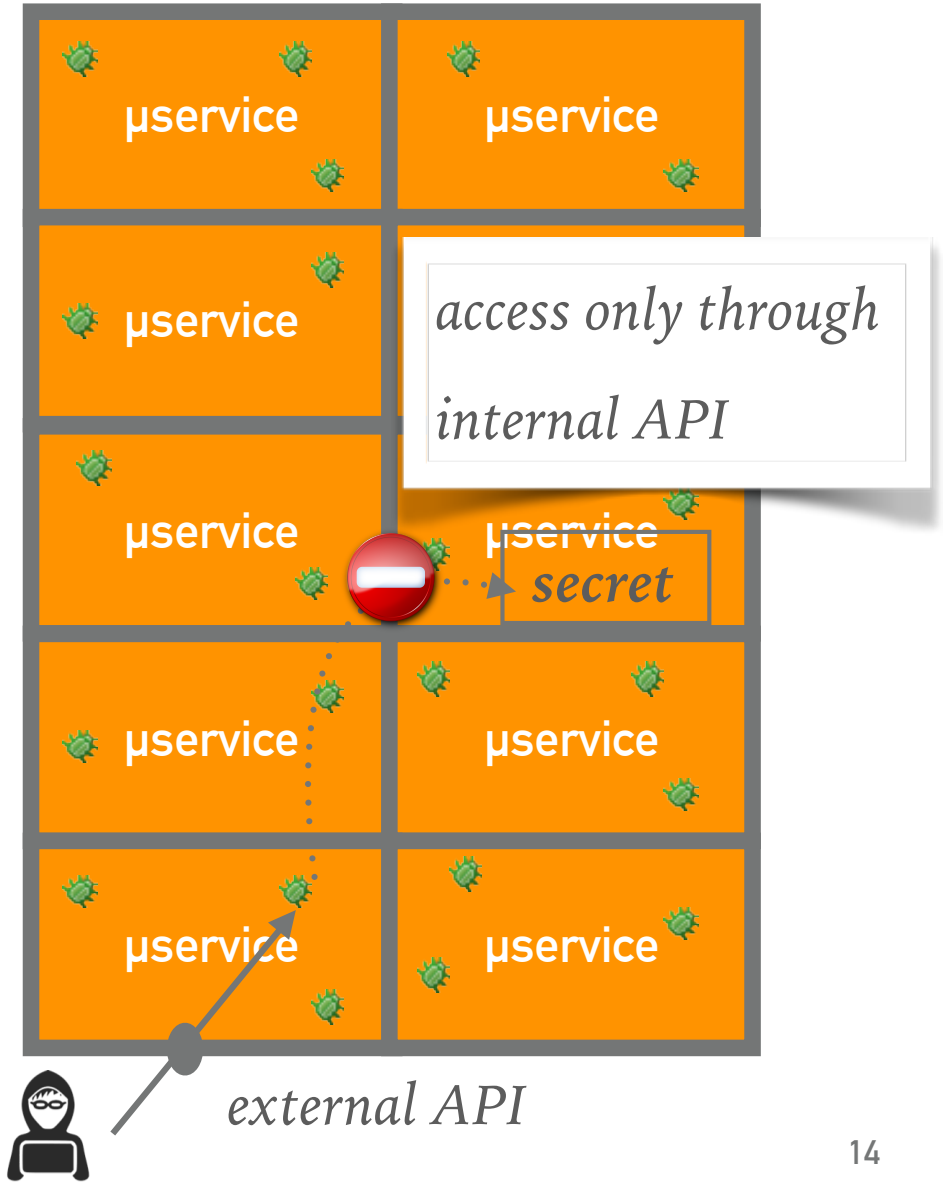
- **SGX:**
  - prevent accesses via privileged / other software
- **Smart adversary:**
  - will exploit bugs inside application code

USE OF MICROSERVICES

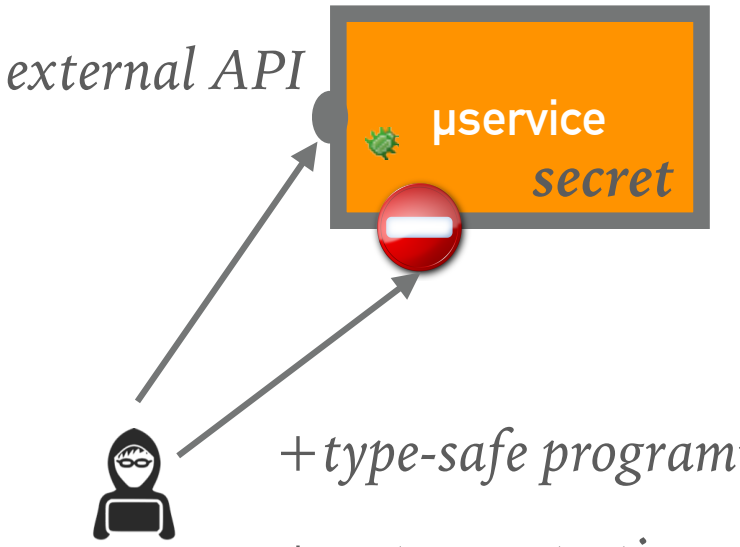
same address space



separate address spaces

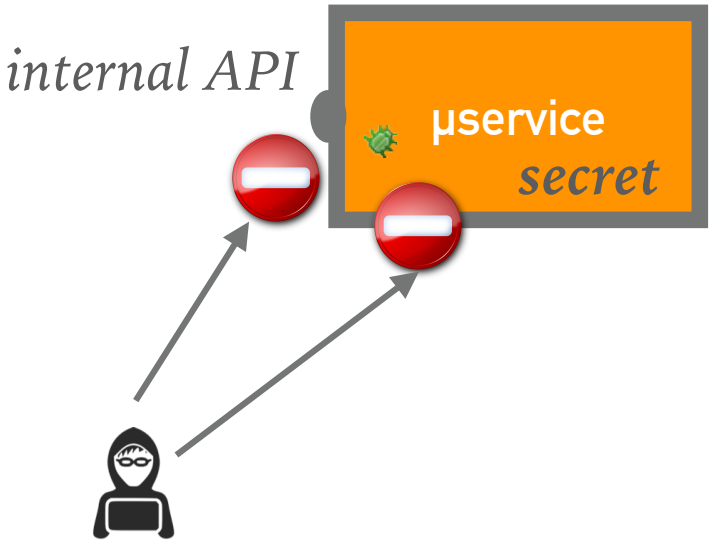


PROTECTING MICROSERVICE APIS



+*type-safe programming languages*

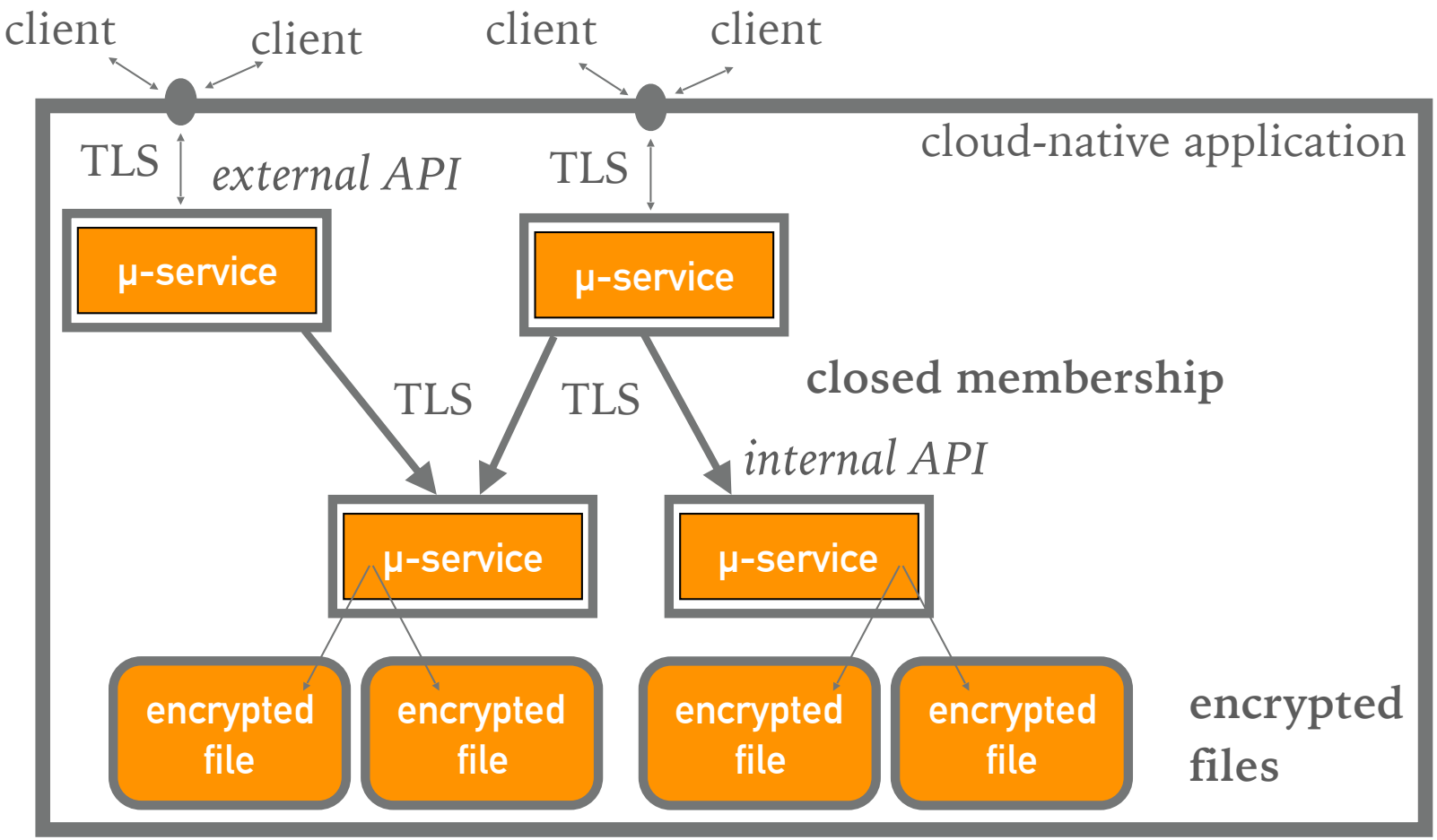
+ *extra protection against attacks*



+*no access to internal APIs*

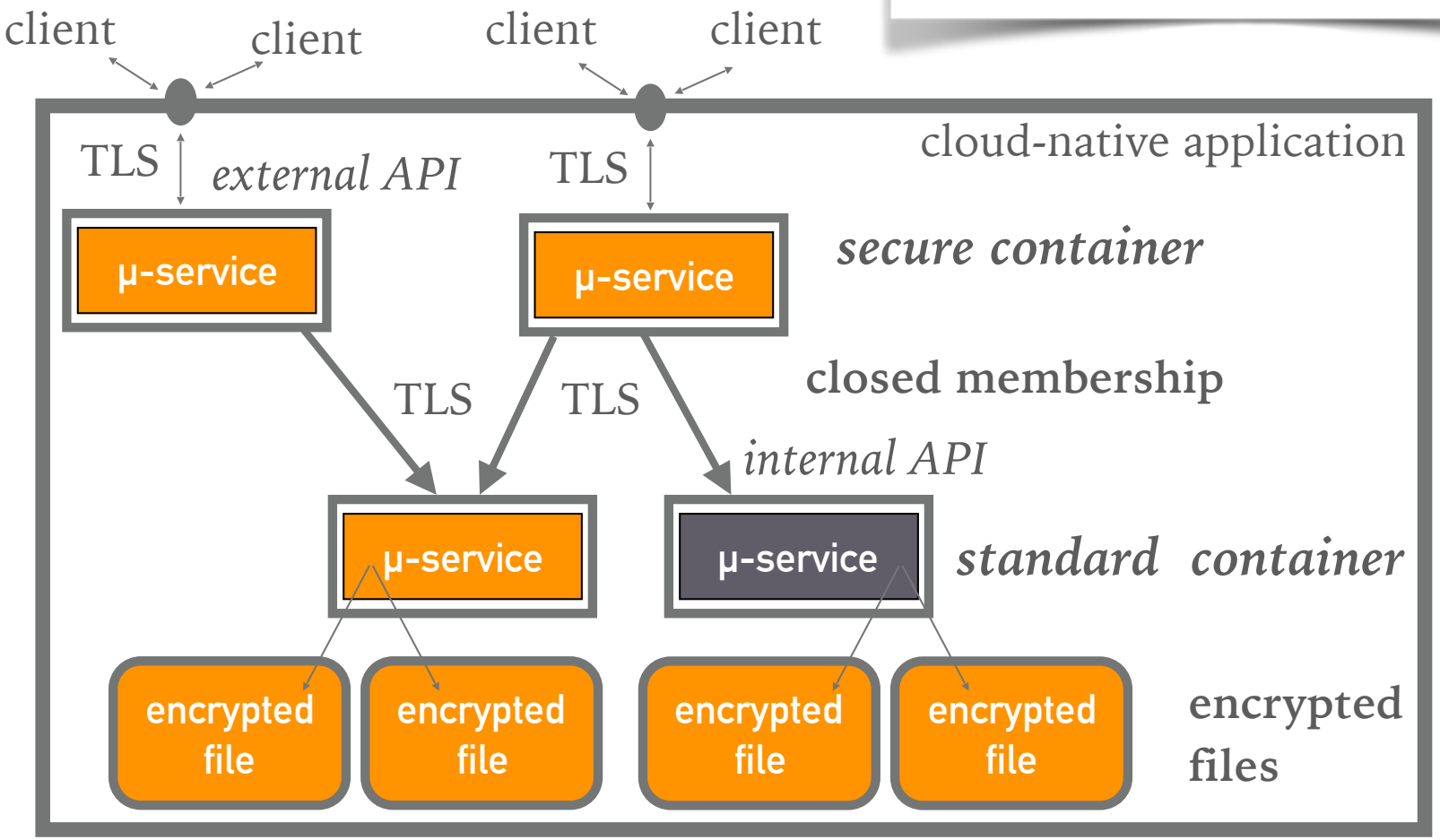
+ *intrusion detection by monitoring internal and external APIs*

SCONE-BASED CLOUD-NATIVE APPLICATIONS



HYBRID APPLICATIONS

not all microservices are critical



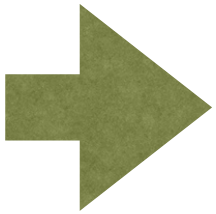
CONTAINER WORKFLOW

- ease of use! -

CONTAINER WORKFLOW

service provider

extended Dockerfile

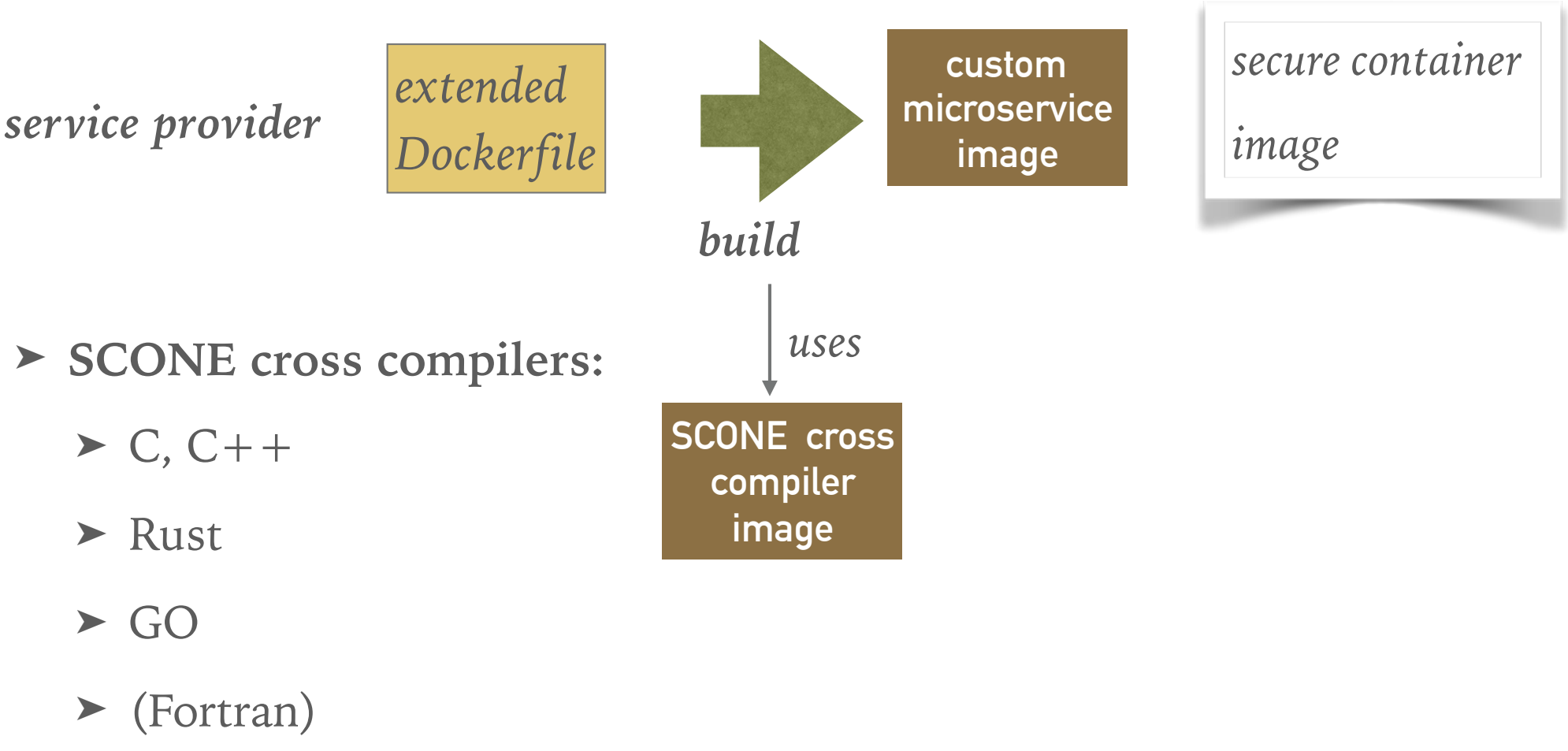


build

custom microservice image

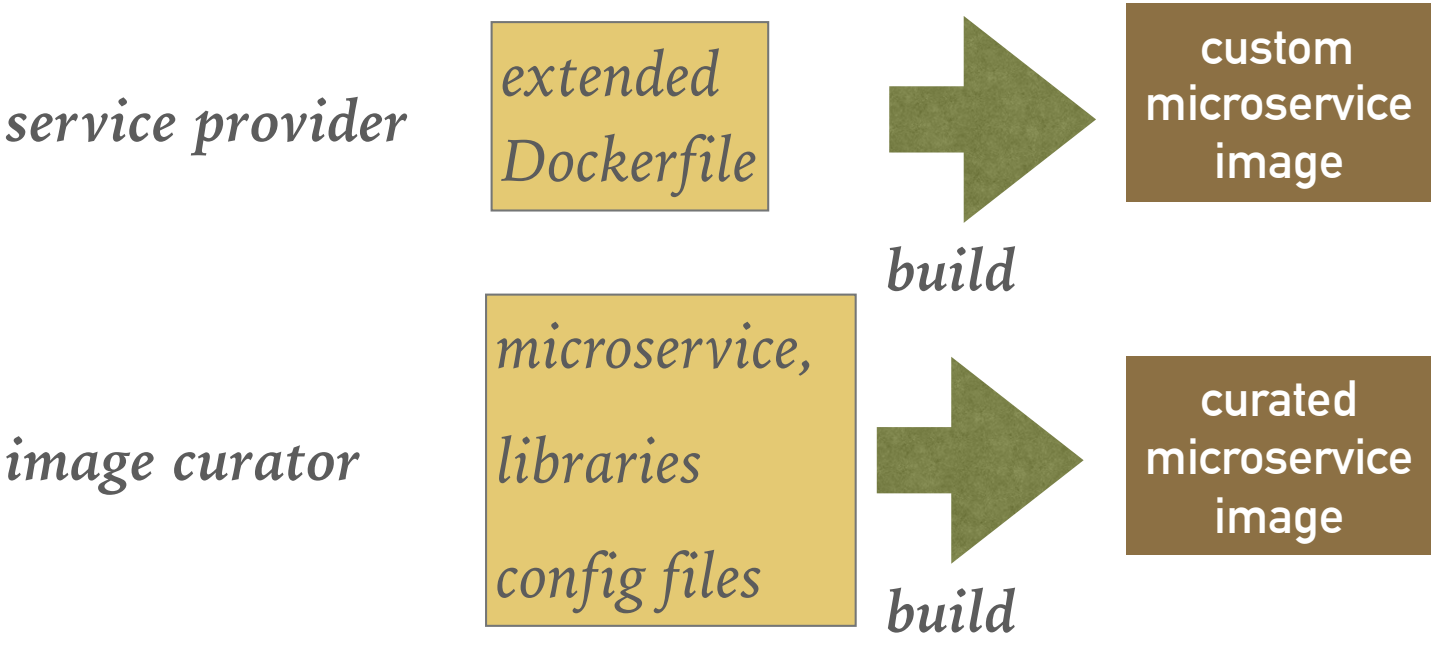
secure container image

CONTAINER WORKFLOW



- **SCONE cross compilers:**
  - C, C++
  - Rust
  - GO
  - (Fortran)
- **Docker**
  - to **build**, ship and deploy images

CONTAINER WORKFLOW



DOCKER HUB

nginx official	5.7K STARS	10M+ PULLS	DETAILS
redis official	3.6K STARS	10M+ PULLS	DETAILS
mysql official	4.1K STARS	10M+ PULLS	DETAILS
mongo official	3.1K STARS	10M+ PULLS	DETAILS

...

hub.docker.com/explore

SCONE CURATED IMAGES (WORK IN PROGRESS)

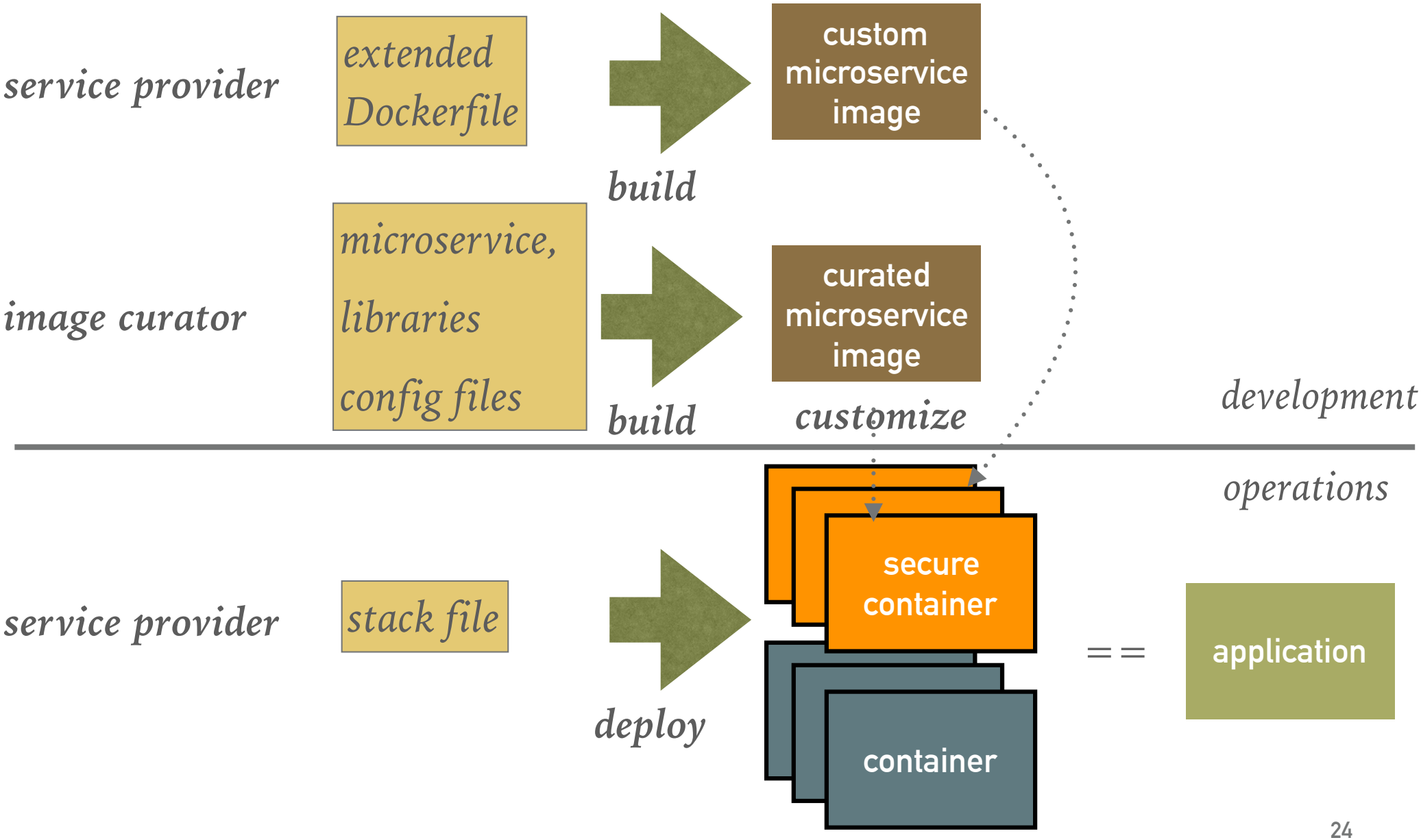
 nginx official	→	*nginx SCONE image*	5.7K STARS	10M+ PULLS	DETAILS
 redis official	→	*redis SCONE image*	3.6K STARS	10M+ PULLS	DETAILS
 mysql official	→	*mysql SCONE image*	4.1K STARS	10M+ PULLS	DETAILS
 mongo official	→	*mongo SCONE image*	3.1K STARS	10M+ PULLS	DETAILS

...

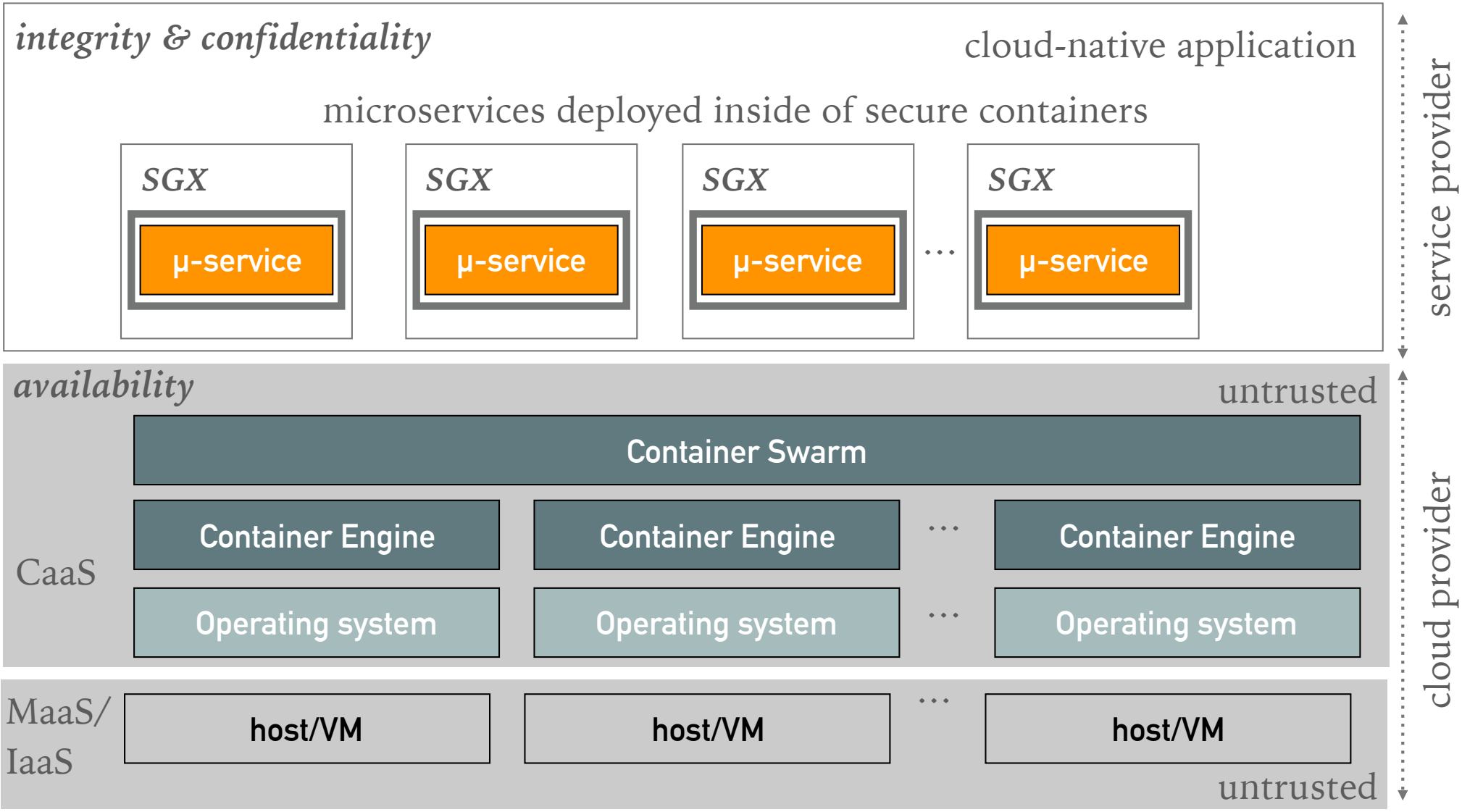
SCONE images are shielded and tuned for SGX

hub.docker.com/explore

CONTAINER WORKFLOW



SERVICE PROVIDER VS CLOUD PROVIDER



COMPOSE EXAMPLE

HOW TO DISTRIBUTE SECRETS?

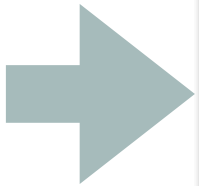
```
mysql-master:
  environment:
    MYSQL_ROOT_PASSWORD: rootpass
    MYSQL_DATABASE: messenger
    MYSQL_USER: messenger
    MYSQL_PASSWORD: messenger
  tty: true
  tty-key: mysecret
  image: mysql
     MRENCLAVE: 0x3394940494
     FSPFKEY:  topsecret
  stdin_open: true
```

- **State of the art:**
  - put passwords in stack / compose file
- **Problem:**
  - Docker engine is not trusted

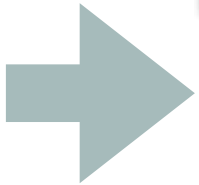
Bad practice to put secrets in compose file!

EXAMPLE: MYSQL

```
mysql-master:
  environment:
    MYSQL_ROOT_PASSWORD: rootpass
    MYSQL_DATABASE: messenger
    MYSQL_USER: messenger
    MYSQL_PASSWORD: messenger
  tty: true
  tty-key: mysecret
  image: mysql
     MRENCLAVE: 0x3394940494
     FSPFKEY:  topsecret
  stdin_open: true
```



split



secrets

```
mysql-master:
environment:
    MYSQL_ROOT_PASSWORD: rootpass
    MYSQL_DATABASE: messenger
    MYSQL_USER: messenger
    MYSQL_PASSWORD: messenger
tty-key: mysecret
MRENCLAVE: 0x3394940494
FSPFKEY: topsecret
```

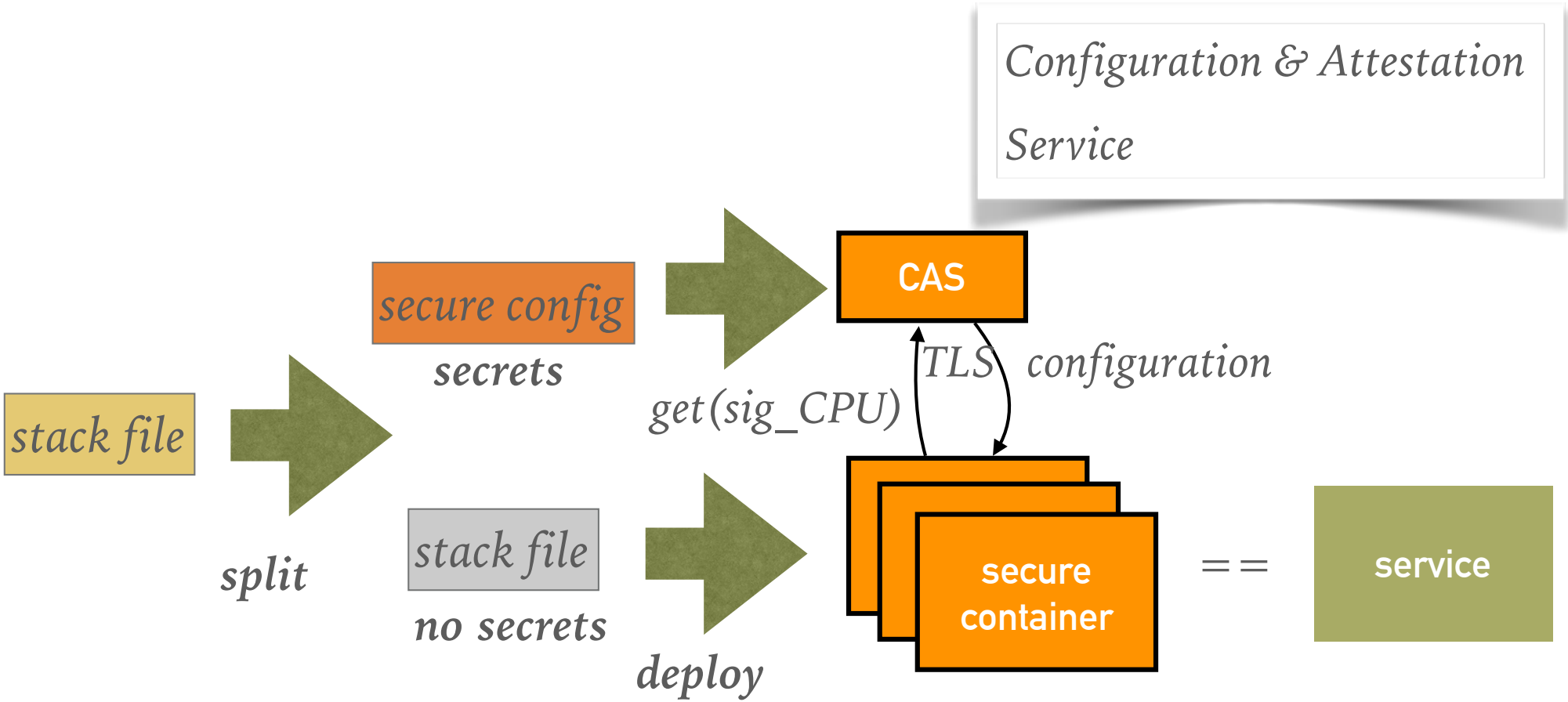
SCONE

no-secrets

```
mysql-master:
  environment:
    APPID:  012345
  tty: true
  image: mysql
  stdin_open: true
```

DOCKER

SCONE: SPLIT STACK / COMPOSE FILE

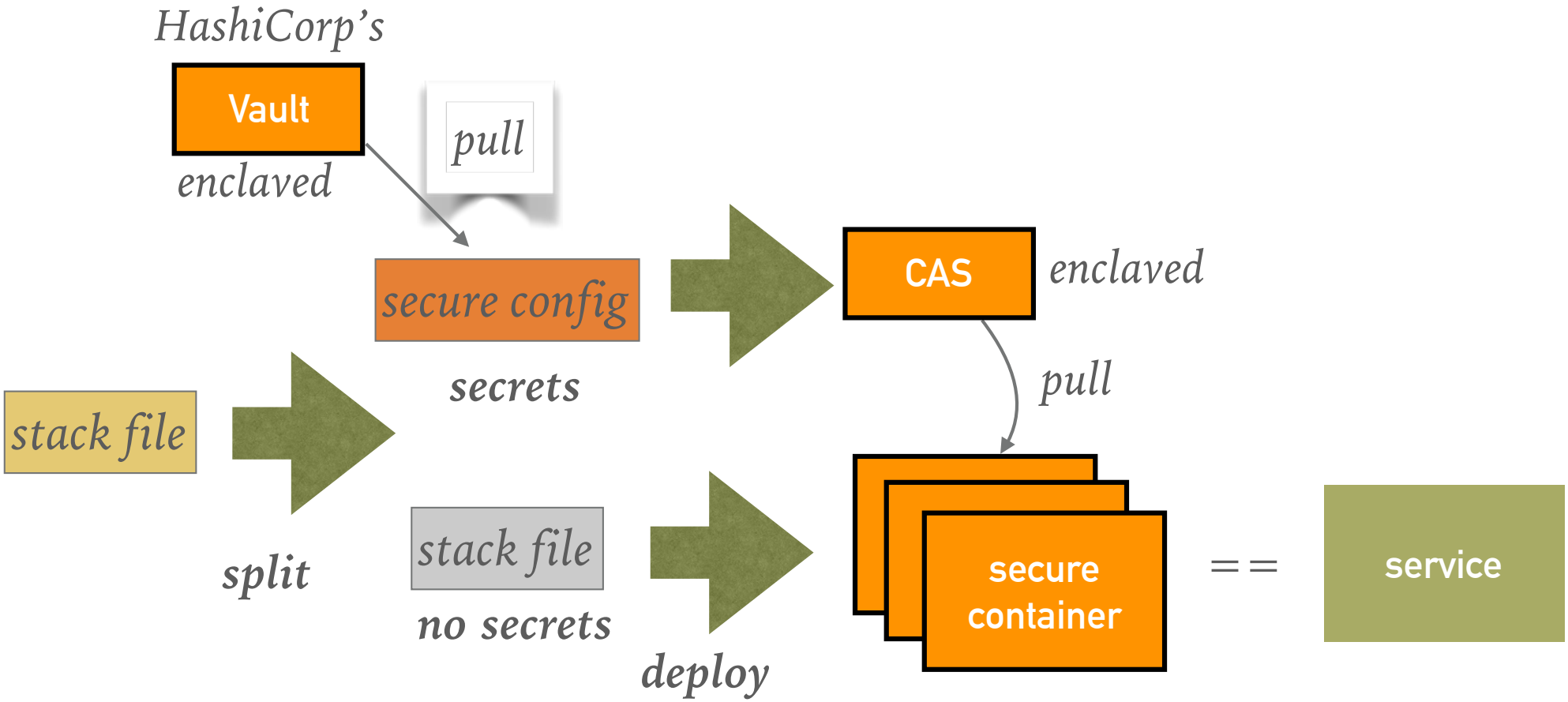


```
mysql-master:
  environment:
    MYSQL_ROOT_PASSWORD: rootpass
    MYSQL_DATABASE: messenger
    MYSQL_USER: messenger
    MYSQL_PASSWORD: messenger
  tty: true
  tty-key: mysecret
  image: mysql
     MRENCLAVE: 0x3394940494
     FSPFKEY:  topsecret
  stdin_open: true
```

PROBLEMS?

- **Stack file**
  - secrets are in the clear
- **Problems:**
  - service administrators might leave company
  - access to secrets by root
- **Approach:**
  - delegate keys to key store like vault

APPROACH: RETRIEVE SECRETS FROM VAULT



EXAMPLE: INTEGRATION WITH VAULT

```
mysql-master:                                   no secrets
  environment:
    MYSQL_ROOT_PASSWORD: $mysql_root_pw
    MYSQL_DATABASE: messenger
    MYSQL_USER: messenger
    MYSQL_PASSWORD: $messenger_pw
  tty: true
  tty-key: $tty_key
  image: mysql
  MRENCLAVE: 0x3394940494
  FSPFKEY: $fspfkey
secrets:
   mysql_root_pw:
       vault: ascii
   messenger_pw:
       vault: ascii
   tty_key:
       vault: AES256
   fspfkey:
       vault: AES256
```

extended stack file

split

```
environment:                                    config file
    MYSQL_ROOT_PASSWORD: xU0932hd…
    MYSQL_DATABASE: messenger
    MYSQL_USER: messenger
    MYSQL_PASSWORD: 9S3jDh1…
tty-key: 0AF1B…
MRENCLAVE: 0x3394940494
FSPFKEY: 3HDJejh…                               secrets
```

```
mysql-master:                                   no-secrets
  environment:
    APPID:  012345
  tty: true
  image: mysql                                  DOCKER
```

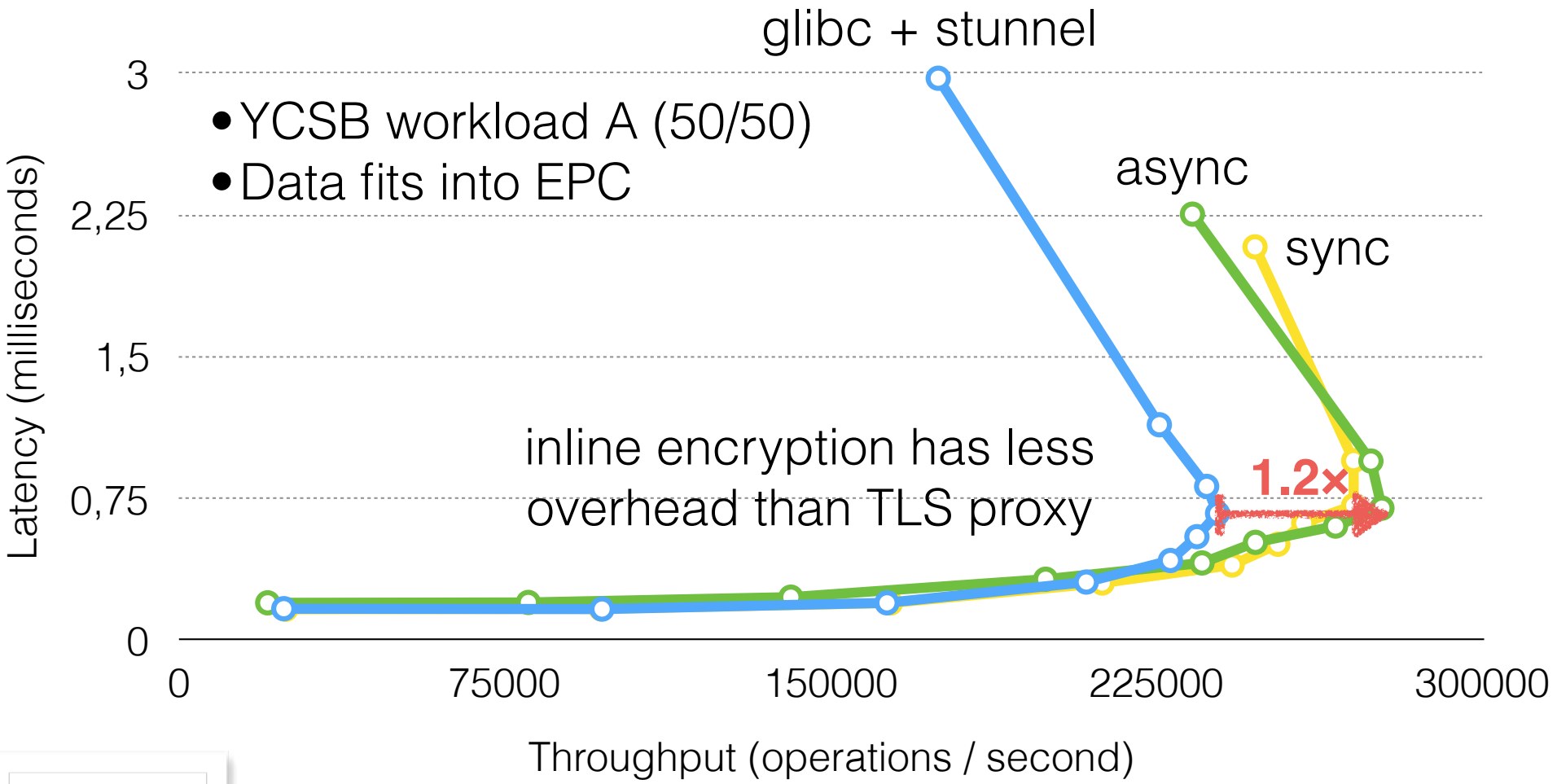
PERFORMANCE

SGX impact

SCONE

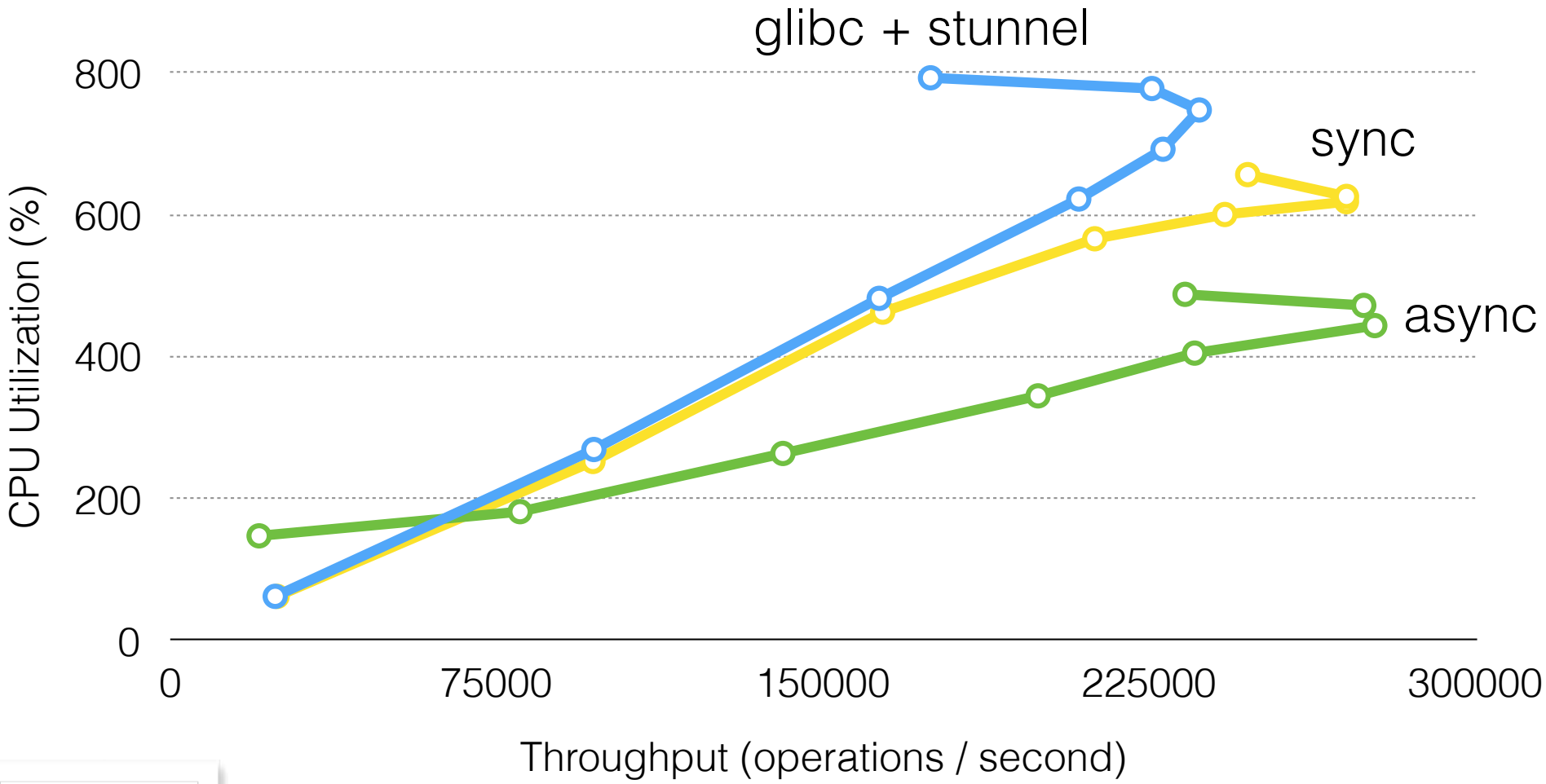
- **Performance optimisations:**
  - **asynchronous interface**: minimise enclave exits
    - syscalls executed by external threads
  - **TLS extensions (new)**
    - support pre-encrypted memory blocks
  - **Autotuner (new)**
    - find „optimal“ values for tuning parameters

Memcached Throughput



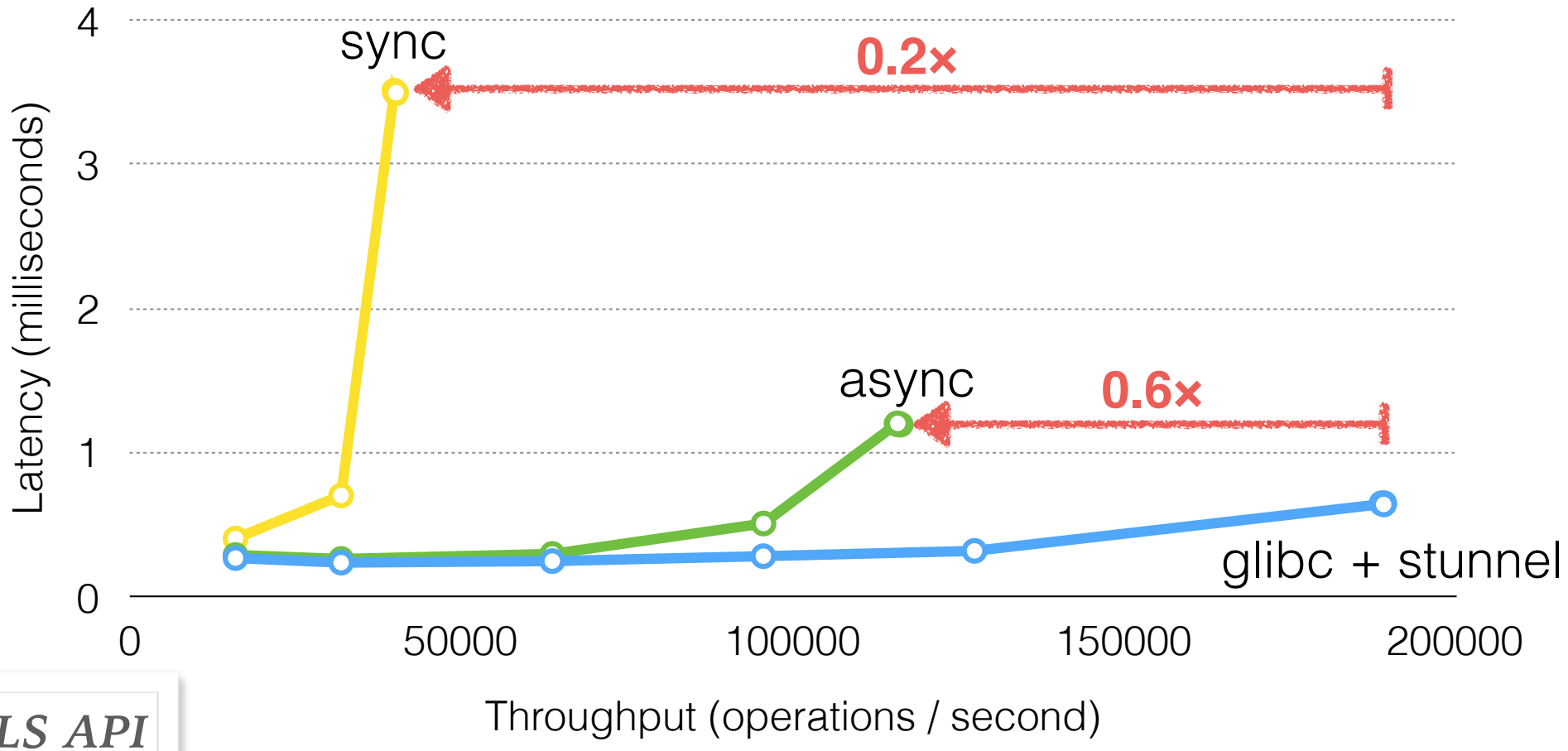
TLS API

Memcached CPU



TLS API

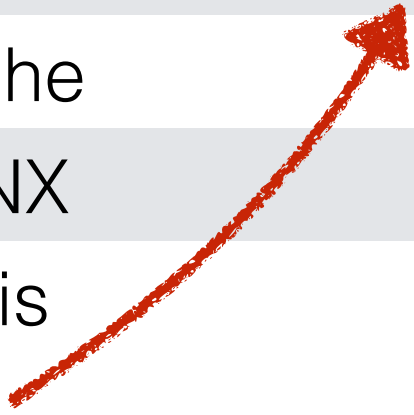
Redis Throughput



TLS API

Performance Overview

Application	Throughput w.r.t. native	
	async (%)	sync (%)
Memcached	120	113
Apache	80	70
NGINX	80	36
Redis	60	20



inline encryption
has less overhead

inline encryption
hurts performance
with single thread

Performance Improvement

Application	Throughput w.r.t. native	
	async (%)	sync (%)
Memcached	120	113
Apache	80	70
NGINX	80	36
Redis	**>80?**	20

current work:
TLS offloading

SCONE SUMMARY

- **ease of use:**
  - look and feel like Docker
- **security:**
  - based on Intel SGX
  - compiler extensions (bounds checker, limit accesses)
- **performance reasonable** (as long as microservice fits in EPC)
  - combine with horizontal scaling if needed
- **practical approach**
    - note: Intel SGX EPC will increase next year…

ADVERTISEMENT

- If you want to evaluate SCONE:
  - **now**: SCONE cross compilers
  - **June**: extended Docker compose
- I'm looking
  - for PhD students and PostDocs
  - for developers who want to join a SCONE startup
- Check out the SCONE documentation

christof.fetzer@gmail.com

```
docker pull sconecuratedimages/sconedocu
docker run -d -p 8080:80  sconecuratedimages/sconedocu
open http://127.0.0.1:8080
```

```
docker pull sconecuratedimages/sconedocu
docker run -d -p 8080:80  sconecuratedimages/sconedocu
open http://127.0.0.1:8080
```