



# Confidential Computing in Distributed Systems

Bachelor's Thesis of

Wenzhe Vincent Cui

at the Department of Informatics
KASTEL – Institute of Information Security and Dependability

| | |
|---|---|
| Reviewer: | Prof. A |
| Second reviewer: | Prof. B |
| Advisor: | M.Sc. C |
| Second advisor: | M.Sc. D |

11. Month 2021 – 02. Month 2022

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself. I have submitted neither parts of nor the complete thesis as an examination elsewhere. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. This also applies to figures, sketches, images and similar depictions, as well as sources from the internet.

**PLACE, DATE**

. . . . . .

(Wenzhe Vincent Cui)

# Abstract

English abstract.

# Zusammenfassung

Deutsche Zusammenfassung

# Contents

# List of Figures

# List of Tables

# 1. Introduction

## 1.1. Motivation

## 1.2. Goal

## 1.3. Research Methodology

# 2. Background

## 2.1. Linux Boot Process

### 2.1.1. Hardware

### 2.1.2. Firmware

### 2.1.3. Boot Loader

### 2.1.4. Kernel

### 2.1.5. initrd

## 2.2. Virtualization and Containerization

### 2.2.1. Virtualization

#### 2.2.1.1. Virtual Machine Manager (VMM)

## 2.3. Containerization

### 2.3.1. Container Manager (CM)

containerd, cri-o

### 2.3.2. Container Runtime (CR)

runc, kata

### 2.3.3. Shim

# 3. Terminology

Expand Terminology

## 3.1. Parties and Roles

- Data Owner
- Program Owner
- Cloud Service Provider

# 4. Cloud Computing

## 4.1. Definition

[1]

### 4.1.1. Characteristics

Explain Cloud Computing Characteristics

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

### 4.1.2. Service Models

Explain Cloud Computing Service Models

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

## 4.2. Benefits

## 4.3. Problems

# 5. Confidential Computing

## 5.1. Trusted Execution Environments (TEEs)

Defined by Confidential Computing Consortium.

### 5.1.1. Properties

- Data confidentiality
- Data integrity
- Code integrity

### 5.1.2. Trusted Computing Base (TCB)

Depending on the specific TEE, it may also provide:

- Code confidentiality
- Authenticated Launch
- Programmability
- Attestation
- Recoverability

### 5.1.3. Models

#### 5.1.3.1. Virtual-Machine-based TEE

- Memory encrypted VM running on top of a VMM

Explain VMM

#### 5.1.3.2. Process-based TEE

Application Splitting:

- **Enclave:** Contains trusted component, resides in encrypted memory.
- **Host:** Interfaces with the OS and propagates I/O from encrypted memory to the rest of the system.

## 5.2. Commodity Hardware Solutions

### 5.2.1. Intel TDX and SGX

### 5.2.2. AMD SEV

### 5.2.3. ARM TrustZone

## 5.3. Measured Boot

# 6. Technical Research

## 6.1. Veracruz

## 6.2. Marble Run

## 6.3. Constellation Kubernetes

## 6.4. Confidential Containers

# 7. Secure Computation Platform

## 7.1. Goal

Provide a platform for

- continuous delivery and deployment (CD)
- deploying services
- running computation tasks and gathering results
- gathering logs and metrics

in a secure environment:

- Data protection (confidentiality and integrity) in every state
- Code integrity
- Automated remote attestation

The goal is not to promise security in every software layer but a trust model which separates the infrastructure layers from the guest application.

The CSP should manage the infrastructure and orchestrate the applications so that the application owner can focus on developing the application instead of worrying about the infrastructure. The orchestration of applications should only include unprivileged actions.

But the security of the application is not part of this platform and should be handled by the application owner. While supporting the developers in deploying their applications the platform should impose as little limitations as possible. The application developer should be able to make own decisions regarding the programming language and the libraries they want to use.

Furthermore, data owners should not blindly trust the CSP. Which means application and data owners should be able to define security requirements and verify them before sending/unlocking sensitive data. These requirements include

- Software running in confidential VMs including initrd and kernel
- Hardware Confidential Computing support
- Firmware (OVMF)

| | Infrastructure as a Service | Platform as a Service | |
|---|---|---|---|
| | | Container | Binary |
| Provides | VMs with CC enabled | Container runtime that runs containers in a CC enabled VM | CC enabled runtime environment |
| Infrastructure & Orchestration managed by | Application Owner | CSP | CSP |
| Development Environment managed by | Application Owner | Application Owner | CSP |
| Notes | • Everything has to be managed by the application owner<br>• Goal is to reduce complexity | • Big attack surface (TCB) if done bad<br>• Much more in control of CSP<br>• Services supporting CC usage can be provided by CSP | • Least control by application owner<br>• Either complex dependency management and language support by CSP<br>• Or language or dependency limitations |

Table 7.1.: An overview over different services models.

## 7.2. Architecture

### 7.2.1. Design Choices

More description for table

#### Containers

- Puts control over development environment into the hands of application owner
- No complex language support and dependency management by CSP
- No language/dependency limitations

#### PaaS Service Model

Lets CSP support application developers

- provide infrastructure and services that help in deployment
  - image building
  - image deployment
  - image registry service
- provide infrastructure and service that help in enabling and verifying CC
  - gathering and sending hardware CC attestation results
  -

#### Kubernetes

Kubernetes is “a portable, extensible, open source platform for managing containerized workloads and services”. It provides some generally applicable PaaS features such as deployment, scaling, load balancing, while being very extensible to enable integration with for example logging, monitoring, alerting, etc. Kubernetes is designed to enable user choice and flexibility, so instead of it being a monolithic platform every solution is optional and pluggable. This enables a wide variety of supported infrastructure platforms.

Why not docker swarm?

## 7.3. Implementation Challenges

## 7.4. Evaluation

# 8. Conclusion

…

# Bibliography

[1] Tim Grance Peter Mell. *The NIST Definition of Cloud Computing*. URL: https://doi.org/10.6028/NIST.SP.800-145.

# A. Appendix

## A.1. First Appendix Section


Figure A.1.: A figure

...