BUILDING CONFIDENTIAL CLOUD-NATIVE APPLICATIONS WITH THE SCONÉ PLATFORM

Christof Fetzer, PhD
christof.fetzer@scontain.com
https://scontain.com
MOTIVATION

➤ **Data-Driven Economy** requires the collaboration

➤ between **non-trusting entities**

➤ **Approach:**

➤ Establish **trust** in the interaction between non-trusting entities
IN CODE WE TRUST WITH SCONE

- establishing trust in untrusted environments between mutually untrusting parties -
DATA-DRIVEN ECONOMY

Example
Use training data
• of a data provider,
• to create AI model,
Protecting the IP of all parties!
Establishing trust with SCONE

**mutual attestation to establish trust**

```
<table>
<thead>
<tr>
<th>microservice</th>
<th>trust</th>
<th>microservice</th>
</tr>
</thead>
<tbody>
<tr>
<td>operates</td>
<td>does not trust</td>
<td>operates</td>
</tr>
</tbody>
</table>
```

Establishing trust

SCONE helps to ensure that communication partners

- run the **correct code base**,
- are **properly initialized**,
- run inside of **TEEs**
Cloud-Native Application

- an application developed and operated using the cloud-native development/operation model

Adapted from Qvarnstrom, Saavedra: „PATH TO CLOUD-NATIVE APP DEV“
## Confidential Cloud-Native Applications

<table>
<thead>
<tr>
<th>Security</th>
<th>Confidential Cloud-Native Application Development</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NEW</strong> data, code, and keys are always encrypted - at rest, in transit, in main memory -</td>
<td></td>
</tr>
<tr>
<td>Focus</td>
<td>Speed to market</td>
</tr>
<tr>
<td>Development Methodology</td>
<td>Agile development, DevOps</td>
</tr>
<tr>
<td>Teams</td>
<td>Collaborative DevOps team</td>
</tr>
<tr>
<td>Delivery Cycle</td>
<td>Short and continuous</td>
</tr>
<tr>
<td>Application Architecture</td>
<td>Loosely coupled, service-based, API-based communication</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Container-centric, portable, scales horizontally, on-demand capacity</td>
</tr>
</tbody>
</table>

[https://scontain.com](https://scontain.com)
Confidential Cloud-Native Application

- cloud-native application
- protect code, data and keys of application
Cloud-Native Application
- an application developed and operated using the cloud-native development/operation model
**CONFIDENTIAL CLOUD-NATIVE APPLICATION**

**SCONE**

- **Proxy**
- **Service**
- **CACHE**
- **DB**

- **clients** (https)
- **http(s)**

- **Kubernetes, OS hypervisor** (untrusted)
- **administrator (root)**
- **cloud provider**
- **adversary (root)**
- **equipment (manufacturer)**

**Confidential Cloud-Native Application**
- cloud-native application
- protect code, data and keys of application

[https://scontain.com](https://scontain.com)
PROTECTION GOALS OF CONFIDENTIAL COMPUTE

➤ Protection of

➤ **Confidentiality**: information is not made available or disclosed to unauthorized individuals, entities, or processes

➤ **Integrity**: information cannot be modified by unauthorized individuals, entities, or processes

➤ **Freshness**: information cannot be replaced by old information by unauthorized individuals, entities, or processes

➤ **Additional Protection goals**:

➤ **Availability**: probability that information is available when it is needed (verifiable via monitoring)

➤ **Durability**: probability that information will survive for one year (verifiable via monitoring)
WHAT INFORMATION TO PROTECT?

➤ Protection of

➤ **Code**, e.g., modern AI programs written in Python

➤ **Data**, e.g., training data to create AI models

➤ **Keys**, e.g., keys used to encrypt databases
WHAT INFORMATION TO PROTECT?

➤ Protection of

➤ **Code**, e.g., modern AI programs written in Python

➤ **Data**, e.g., training data to create AI models

➤ **Keys**, e.g., key used to encrypt database

➤ **Example:** *Cannot protect encryption key in native execution*

➤ *MariaDB* supports encryption of database

➤ encryption key is stored in configuration file

➤ configuration file protected via access control:

➤ i.e., can be read and written by *MariaDB* (user) as well as any root (＝privileged) user
PROTECTION WITH SCONE

➤ Protection of

➤ **Code**, e.g., modern AI programs written in Python
➤ **Data**, e.g., training data to create AI models
➤ **Keys**, e.g., key used to encrypt database

➤ **Example**: Confidential Cloud-Native Application with SCONE

➤ *MariaDB* encrypts database and runs **in SGX enclave**
➤ encryption key is stored in configuration file encrypted/
decrypted by SCONE inside of MariaDB enclave
➤ SCONE configuration attestation service ensures that only
this MariaDB can access the encrypted configuration file

https://scontain.com

Confidential Cloud-Native Applications
HOW?
CONFIDENTIAL CLOUD-NATIVE APPLICATION DEVELOPMENT

Methodology:
- Service Based
- API-Driven
- Confidential Compute
- Containers
- DevOps

Tools:
- SCON IDE, git, …
- SCON
- Kubernetes, Docker

Development

Deployment / operations
INTEGRATION IN CI/CD PROCESS

Development

Service Based
API-Driven
Confidential Compute
Containers
DevOps

Deployment / operations

CI/CD
Build (SCONE)
Key Management (SCONE)
Encrypted Execution (SCONE)

SCONE

https://scontain.com
Confidential Cloud-Native Applications
WE SUPPORT MOST POPULAR PROGRAMMING LANGUAGES

Service Based
API-Driven
Confidential Compute
Containers
DevOps

Development

Source (e.g., git) → Build (SCONE) → Key Management (SCONE) → Encrypted Execution (SCONE)

SCONE

RUST
Python
C
C++
Java
node
Fortran

Deployment / operations

SCONE
Confidential Cloud-Native Applications
MICROSERVICE

microservice
• focus on a single aspect
• microservices are small, autonomous services that work together

REST API
http(s)

Cloud-Native Application Component
files
Protecting Confidential Cloud-Native Applications without source code changes
SCONE supports the development & operation of confidential cloud-native applications
info@scontain.com
https://scontain.com
https://sconedocs.github.io/