Training

System Description

Abstract
This is the System Description (SysD document) for the Training System according to the Eclipse Arrowhead documentation structure.

Contents

1 Overview

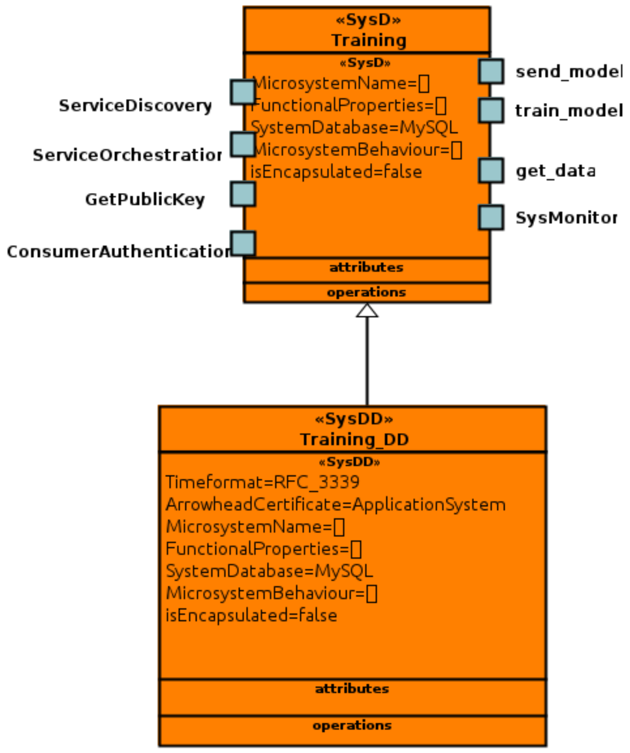

Figure 1: Block Diagram representation of the AI Tool microsystem

The rest of this document is organized as follows. In Section **??**, we reference major prior art capabilities of the system. In Section 1.1, we describe the intended usage of the system. In Section 1.2, we describe fundamental properties provided by the system. In Section 1.3, we describe delimitations of capabilities ofn the system. In Section 2, we describe the abstract service functions consumed or produced by the system. In Section 3, we describe the security capabilities of the system.

1.1 How This System Is Meant to Be Used

This system is used to do edge caching and local inference in order to train the AI model and send its tensor to the AI-tool.

1.2 System functionalities and properties

1.2.1 Functional properties of the system

- The system can receive the preprocessed data
- The system can change the weights and bias of a model
- The training is done periodically
- The system can store and send the model

1.2.2 Configuration of system properties

1.2.3 Data stored by the system

The IP packets are stored as CSV in a server close to the training system in order to train a model and store only the weights on the edge in order to execute the model in the IA-tool.

1.2.4 Stateful or stateless

Stateful: All the data received is stored for a specific period of time in order to train the model with a bigger quantity of packets, so the attacks like DDoS can be seen too. The model weights are also stored and when it is fine-tuned, the new model weights overwrite the old version.

1.3 Important Delimitations

The system can be delimited by the computing power of the CPU/GPU. With light models and high-performance runtime for on-device AI such as LiteRT this can be solved. Nevertheless, reducing the size of the model may be a challenge and a field for research in order to have and accurate detection of intrusion. Also, having enough data to train or fine-tune a model may be an issue at the beginning. There should be some monitoring in order to avoid overfitting.

2 Services

2.1 Produced service

- ”send model”: The service sends the tensor of the model to the AI-tool system.
- ”train”: This service changes the weights of the model training with the data received and stored for a period of time.
- ”SysMonitor”: This service provides real-time log messages to an external monitoring system. It is based on the specifications detailed in the MicrosystemMonitor SD document.

2.2 Consumed services

- ”send data”: The system receives the preprocessed data sent by the Preprocess system and stores it in order to use it while executing the service ”train”.
- ServiceDiscovery: Essential for communication with the registry.
- ServiceOrchestration: Coordinates the system.
- GetPublicKey: Provides the necessary authorization mechanisms.
- ConsumerAuthentication: Manages the authentication of external entities wishing to access service data.

3 Security

- The Training system utilizes secure protocols such as:
  - TLS (Transport Layer Security): Provides secure communication over the network, protecting data integrity and privacy during transmission.
  - MQTT over TLS: For low-latency, lightweight communication in IoT and edge environments, ensuring data is transmitted securely.
  - HTTP/HTTPS: For web-based communication, with HTTPS ensuring secure communication via encryption.
- The system ensures data protection through:
  - Encryption: Data in transit is encrypted using strong algorithms (e.g., AES) to prevent unauthorized access.
- The system performs strict authorization checks before providing services, based on:
  - OAuth 2.0: Ensures that only authorized clients can access services, with tokens being issued and validated before service access.
  - Arrowhead Authorisation System: This checks the legitimacy of service requests within the Arrowhead ecosystem, ensuring that services are only consumed by authorized actors.

For Arrowhead certificate profile see github.com/eclipse-arrowhead/documentation

4 References

5 Revision History

5.1 Amendments

Revision history and Quality assurance as per examples below

No.	Date	Version	Subject of Amendments	Author
1	2020-12-05	X.Y.Z		Tanyi Szvetlin
2	2021-07-14	X.Y.Z	Minor updates	Jerker Delsing
3	2022-01-12	X.Y.Z	Minor updates	Jerker Delsing

5.2 Quality Assurance

No.	Date	Version	Approved by
1	2022-01-10	X.Y.Z	