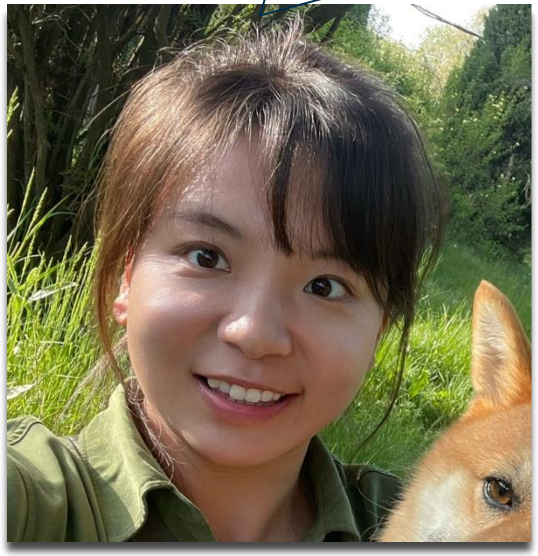
RAdaptSQ:
Real-Time AI-Planning and Environment-Aware Self-Adaptation to Optimize Security and QoS.

I am the speaker



Lin Cui
Karlsruhe Institute of Technology
Karlsruhe, Germany
lin.cui@kit.edu

Raffaela Mirandola
Karlsruhe Institute of Technology
Karlsruhe, Germany
raffaela.mirandola@kit.edu

Legend
Problem / question / challenge
Solution / idea / Anwser
Other content

Background

Microservice application

- Decompose monolithic functionality into **independent, single-purpose services**.
- **Simplifies updating**, scheduling, scaling, and integration
- **Widely adopted across industries**: 74% of respondents already use a microservices architecture; the remaining 23% plan to (Gartner 2023).

SErvice-based E-Food Application(SEFA)

- The prototype was implemented by RAMSES.
- Ordering: manages carts and finalizes orders
- Restaurant: handles restaurant listings and menus
- Payment proxy: connects to a third-party payment provider
- Delivery proxy: arranges food delivery

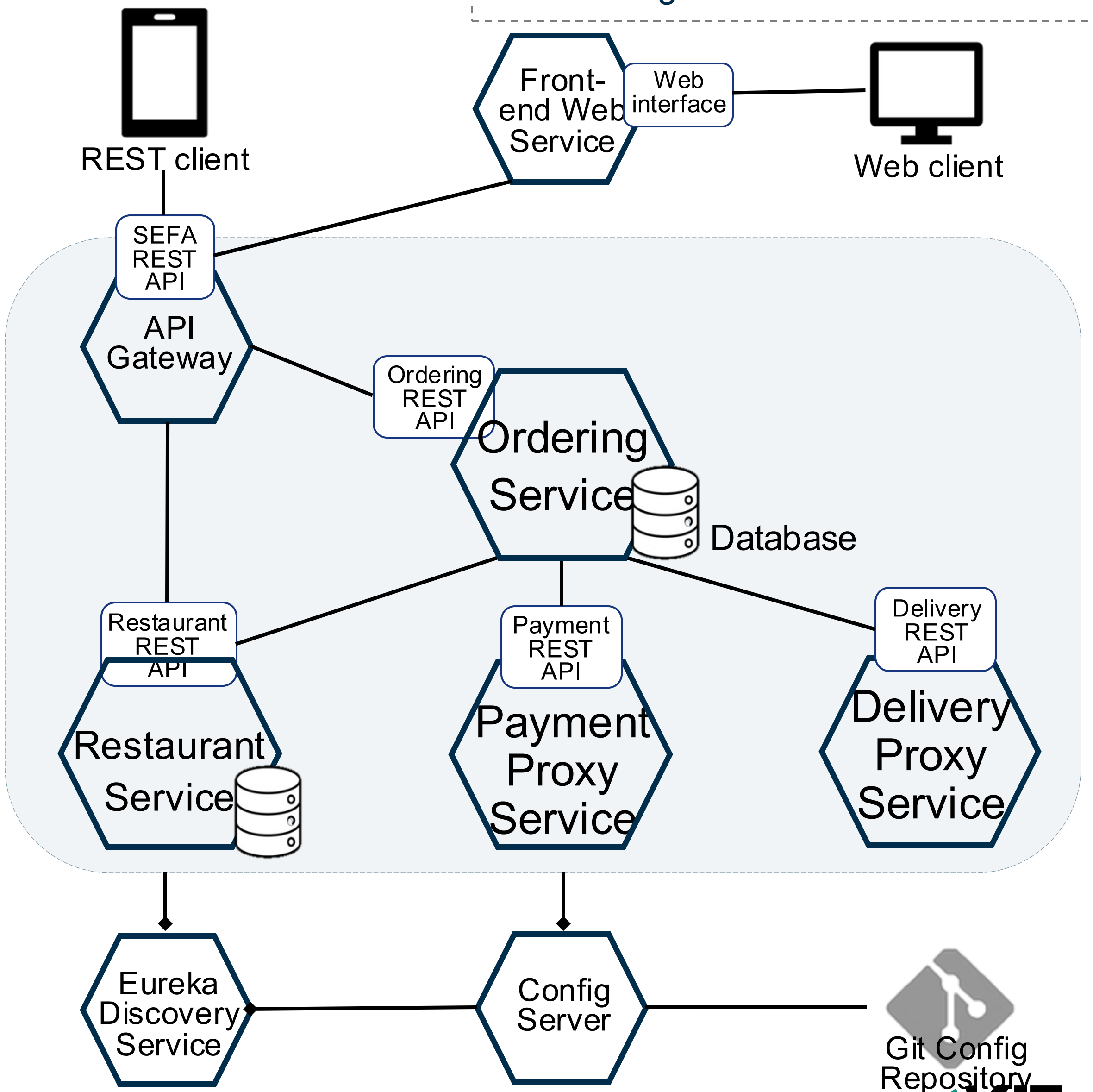


Figure. SEFA architecture

Characters

Lin Operations Engineer

- 9 years of experience
- Owns a Border Collie
- Works hard to earn a living and support the family

Characters

Some things are fictional

Danjiao Attacker

- Evil dog, known as DJ in the community
- Passionate about breaking
- Very smart, currently learning computer skills

Characters

Raffaela Leader

- Nearly 30 years of experience
- Highly competent and meticulous

Problem Statement

Some things are fictional

The uncertainties (order volume spike, version changes, attacks…) are only known during operation

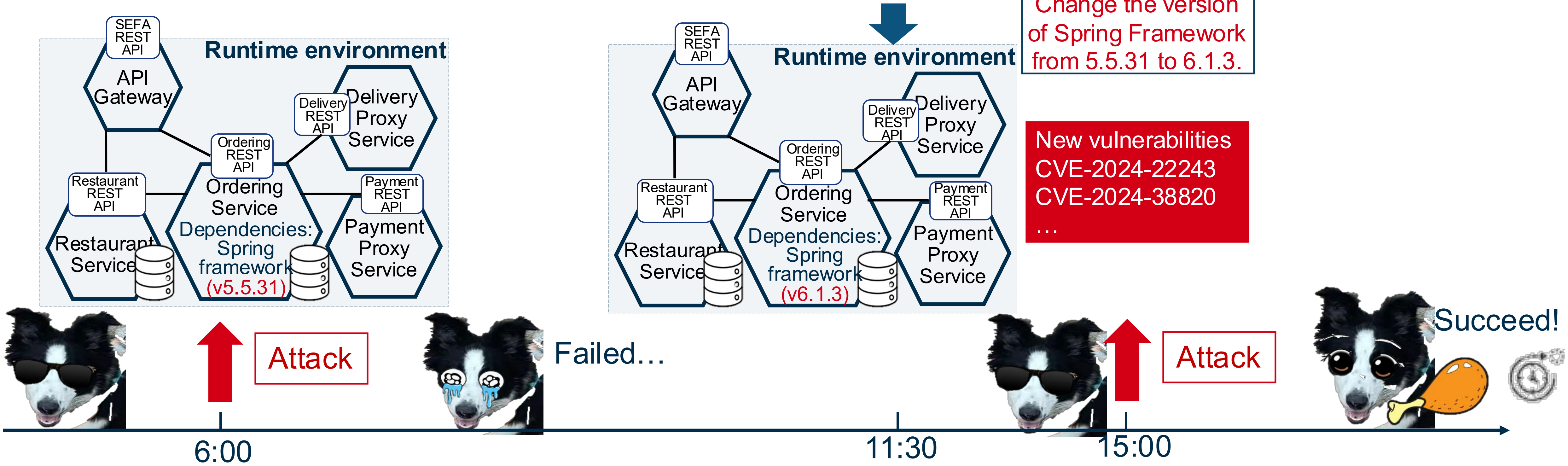


Figure. How runtime uncertainties in microservice applications cause security drift

Rapid fluctuation of the attack surfaces

- Change during runtime introduces new threats, vulnerabilities, the attack surfaces change quickly

Security Drift

- Static security controls with fixed strategies become outdated and ineffective shortly after deployment

Problem Statement

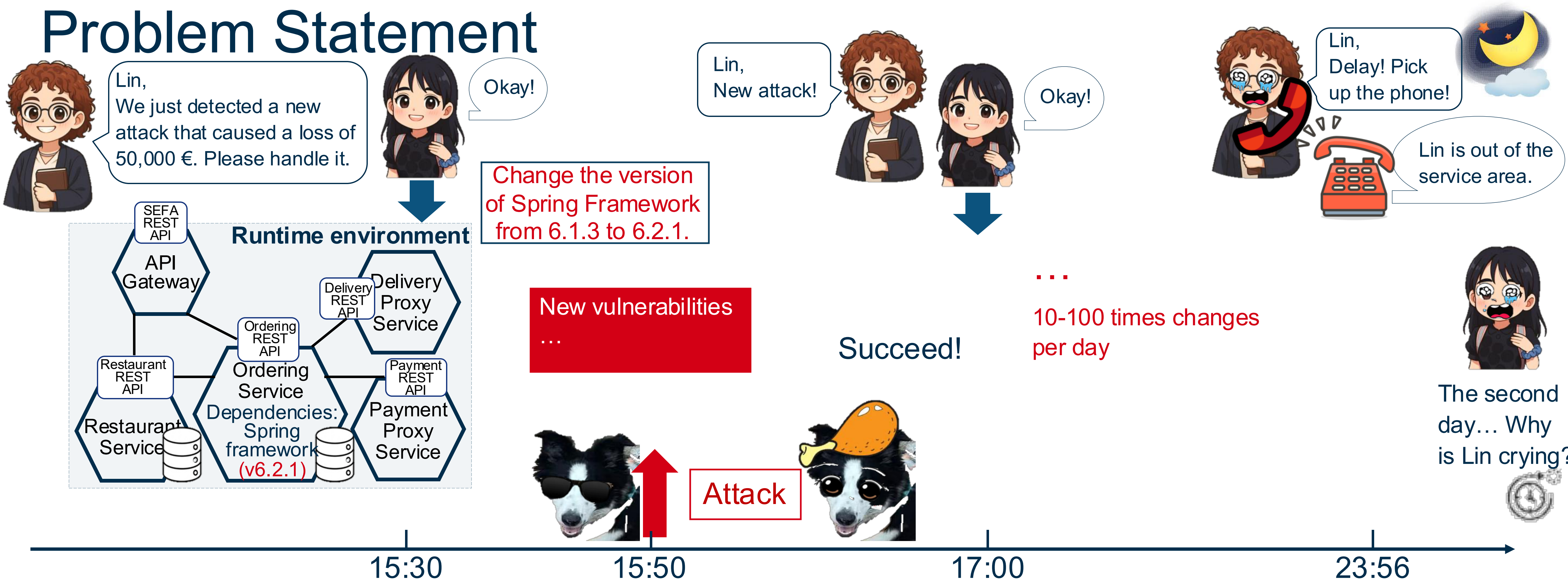


Figure. How runtime uncertainties in microservice applications cause security drift

Rapid fluctuation of the attack surfaces

- Change during runtime introduces new threats, vulnerabilities the attack surfaces change quickly

Security Drift

- Static security controls with fixed strategies become outdated and ineffective shortly after deployment

Rapid Security Drift

- Manual Maintenance unrealistic, can never keep up with the pace of change.

Problem Statement

Is there a way to **automatically handle uncertainties** during an app's runtime while maintaining system performance and security?

Envisioned Solution

Assumption: SAS won't introduce new attack surfaces to the microservice app

Why Self-adaptive System (SAS)?

- **Self-adaptation** is **one prominent approach to deal with uncertainty** and business continuity **autonomously**
- Key idea: let system **gather new knowledge at runtime to resolve uncertainties**, reason about itself, its context and goals, and adapt to realize goals

What is SAS?

- **External principle:** A self-adaptive system is a system that can **handle uncertainty** in its environment, itself and its goals **autonomously** (or with minimal human interference)
- **Internal principle:** A self-adaptive system comprises two distinct parts:
  - 1. **Managed system** interacts with the environment and fulfils business functions.
  - 2. **Managing system** monitors and adapts the managed system to handle adaptation concerns and achieve self-adaptation.

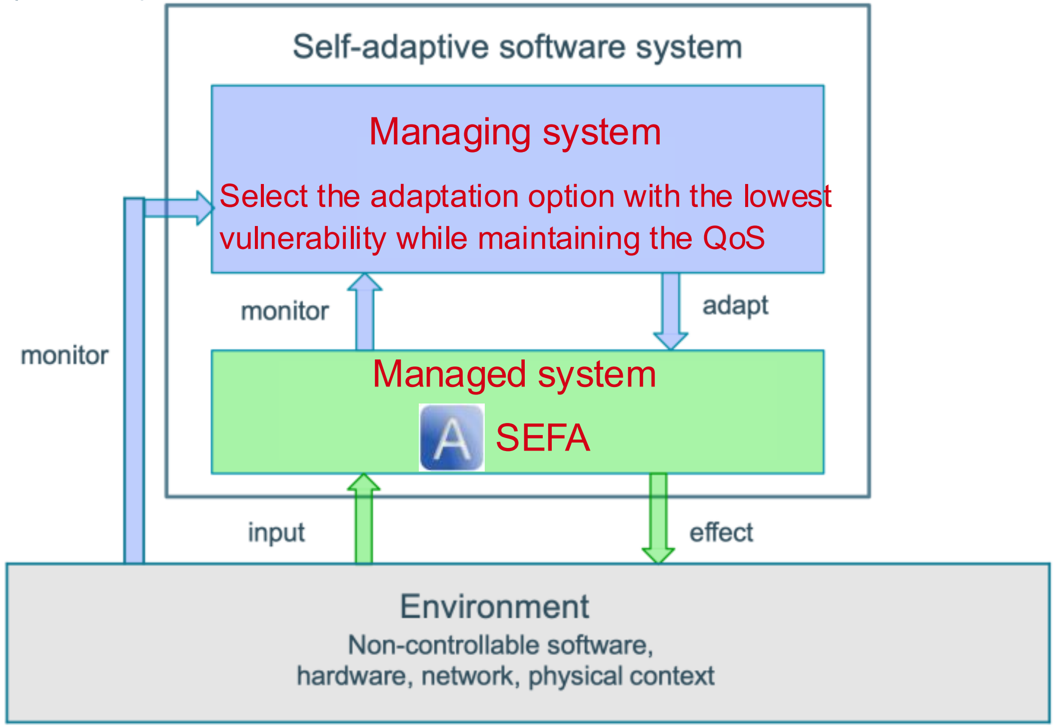


Figure. Self-adaptive system

Gap in the Existing Solution: AQUA [7]

- Representative work to make security enhancement an adaptation objective for microservice Apps by integrating security assessment into an adaptive MAPE-K loop.
- **Security assessment performed offline** and **remains static at runtime.**
- **Cannot handle fluctuating attack surfaces.**

Research Questions

RQ1: Modeling the App's Endogenous Configuration and its Exogenous Environment

How to efficiently model the managed microservice app's internal configuration and external environment within the Knowledge module?

- Update in a timely manner, consistently reflect changes, and the system's latest runtime state.

Goal: Maintaining a model within the Knowledge module that consistently reflects the managed system and the environment

Basis. Model in the Knowledge module serves for adaptation desicion making

RQ2: Dynamic Assessment and Selection the Optimal Adaptation Option

How can the MAPE-K loop dynamically assess the runtime security of the system based on the latest internal and external states?

- Meet the stringent time requirements of microservice environments.

Goal: Spotting the optimal option that optimize the app's security while maintian acceptable QoS

Contribution: Conceptual Framework of RAdaptSQ

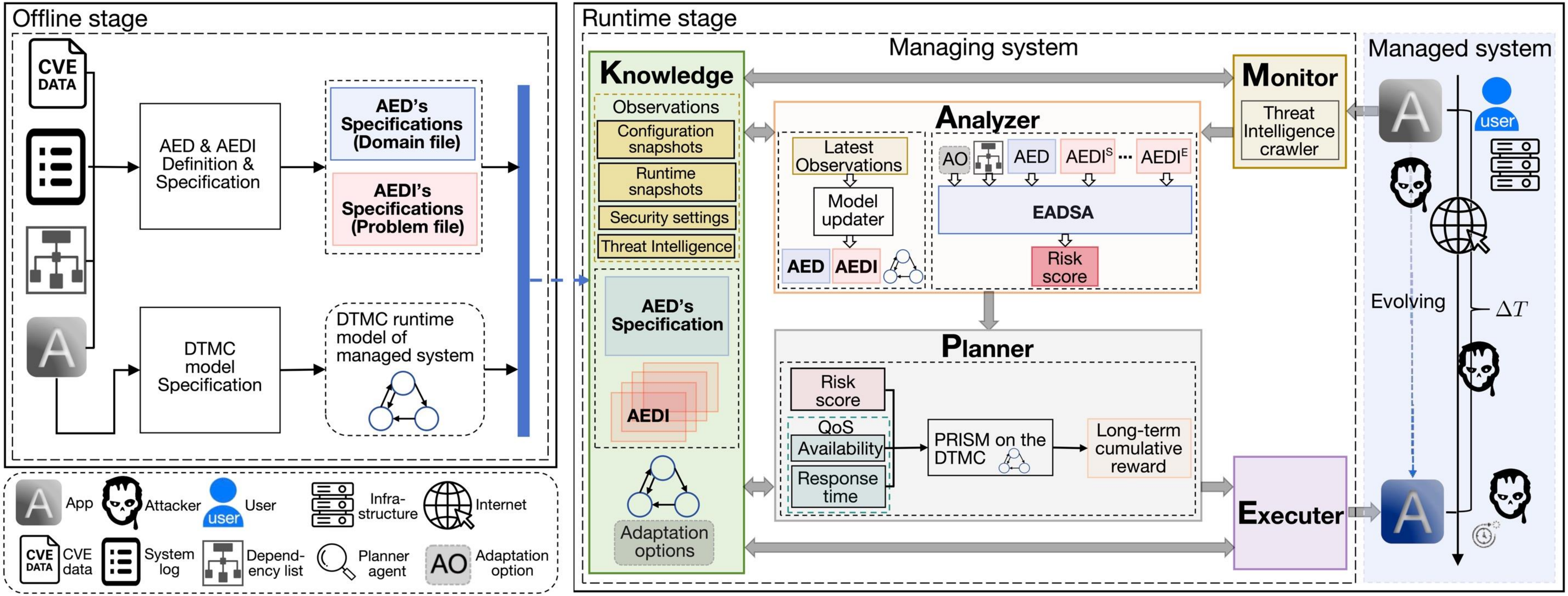


Figure. Conceptual framework of RAdaptSQ

Abbreviation

AED: Adversarial Environment Domain; AEDI: Adversarial Environment Domain Instance; EADSA: Environmental-Aware Dynamic Security Assessment

Ideas for RQ2 (Dynamic Assessment)

RQ2

- How can the MAPE-K loop dynamically assess the runtime security of the system based on the latest internal and external states?

Brainstorm for Potential Solution: Security assessment –> Attack Paths planning

Design a method to assess the security of the system state with **rapidly changing runtime conditions**?

- Meet the **strict time requirement**
- **Dynamic**
- **No human interaction**

- **Attack Path (AP):** detail the step-by-step attack actions and prerequisites enabling an attacker to progress from the initial state to the target.
- **One extensively used method for analyzing system security.**

Can we use AP at runtime?

Traditional formulations, tooling are used **offline, statically**.

- **Meet strict time requirements** ✗
- **Incremental/partial updating** ✗

If we relax the formalism, an **AP is simply an ordered sequence of actions** that achieves a given attack objective.

AI planners help robots **automatically plan** a **sequence of actions in real time** to accomplish a specific task, such as placing a book on a table.

Can we use an AI planner specifically to generate attack paths?

Envisioned Solution

What is AI planner?

- An AI planner is a system that **automatically generates a sequence of actions** to **achieve a specific goal** from a given starting state.
- Features:
  - **Real-time planning:** can operate in real-time.
  - **Dynamic:** generated action sequences (plans) adjust automatically when the environment changes.
  - **Automation:** Reduces human effort in complex decision-making.
- The AI planner is extensively used in robotics, autonomous vehicles, smart homes, game AI, logistics, scheduling, and etc.

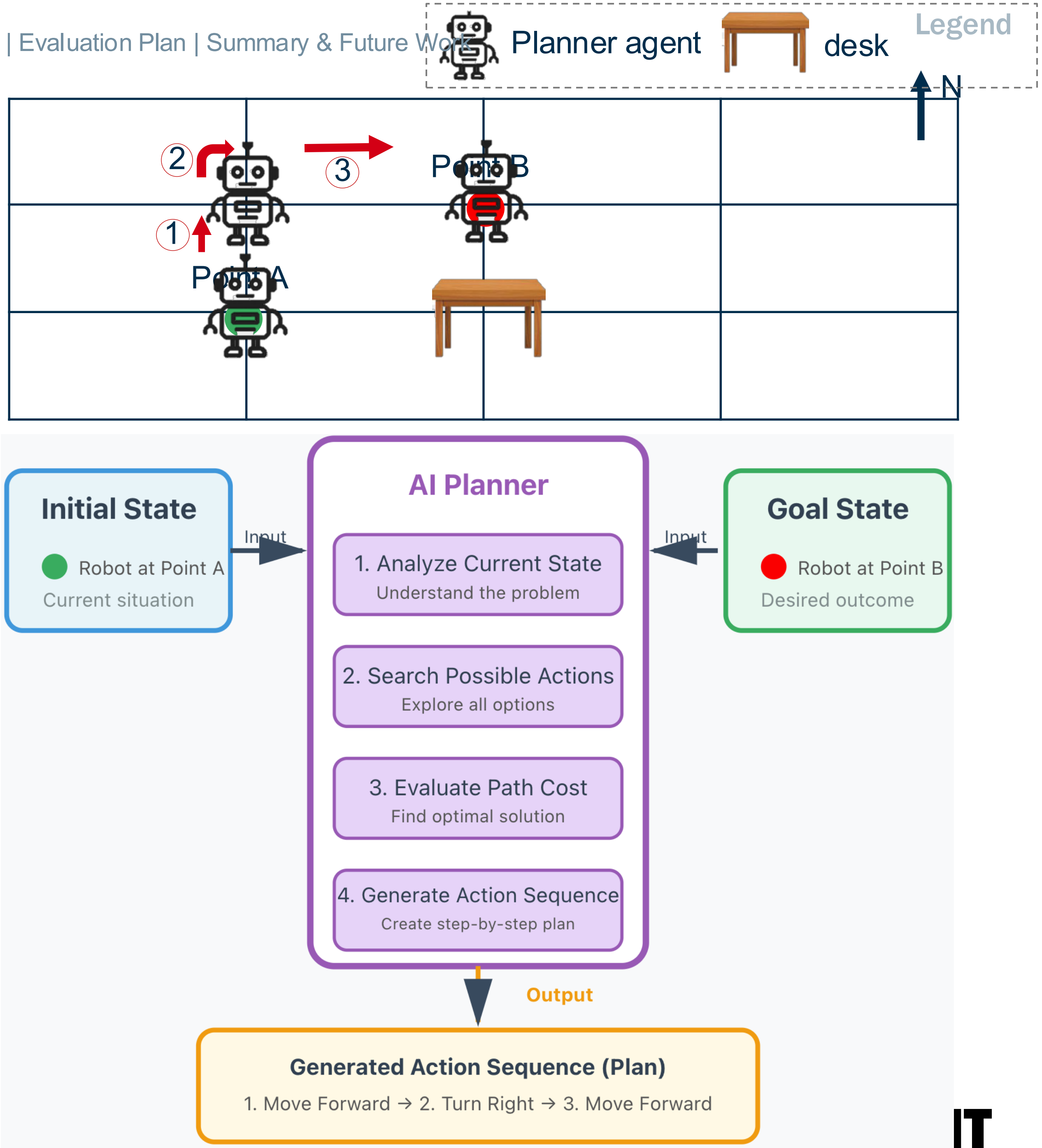
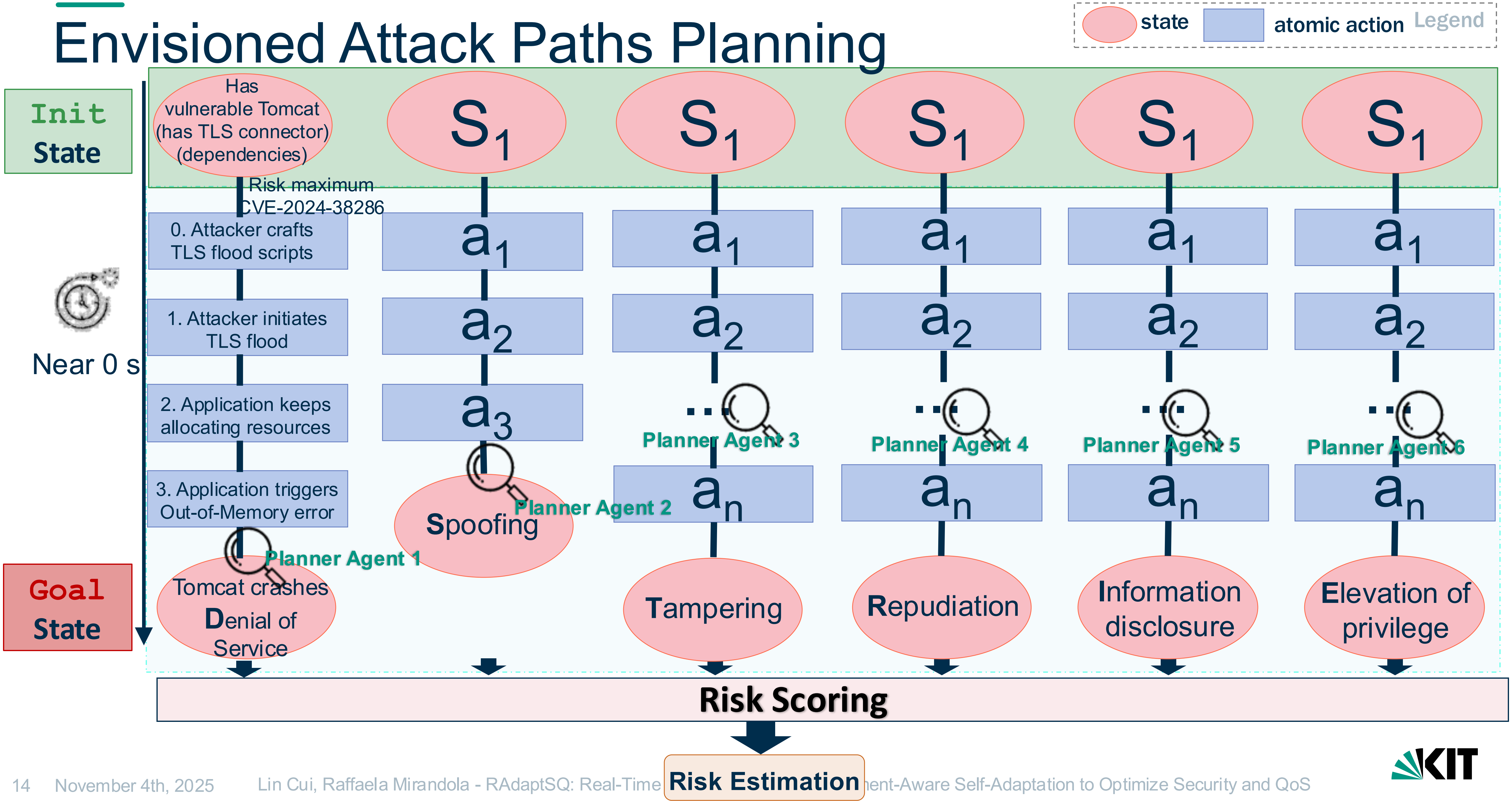


Figure. AI planner

Envisioned Attack Paths Planning



Problem Statement

Are there any readily usable AI planners?

If so, how to **use AI planners**?

How to **interpret the system's initial state and the target goal state** for AI planners?

Envisioned Solution

Why Planning Domain Definition Language (PDDL)?

- PDDL is a **standardized description language for automated AI planning**.
- By standardizing the representation of planning problems, **PDDL enables most AI planners**, such as Metric-FF and Fast Downward, **to interpret and solve planning problems**.

How to use PDDL?

- The PDDL standardizes planning problem description via:
  - Problem file, instantiates the domain for a specific scenario. It lists concrete objects, specifies the initial state as a set of grounded predicates, and desired goal states to be reached.
  - Domain file, defines the model of the environment and system by specifying types, predicates, and actions.
- **A PDDL planner receives these files as input and searches for an action sequence that transforms the initial state into one satisfying the goal.**

```
(define (domain AED)
(:requirements :strips :typing ...)
(:types app lib...)
(:predicates  (ddos ?x - app)(mem-ex ?x)
              (has-lib ?x - app ?y - lib ) ...)
(:functions (version ?x)...)
(:action app-becomes-unresponsive
 :parameters (?x - app)
 :precondition (and (mem-ex ?x )...)
 :effect (and (ddos ?x)))...)
```

Types

States and relationships of typed objects

Numerical features of objects.

Executable actions of attackers, applications, users, or infrastructures

Figure. Example of PDDL domain file

```
(define (problem AEDI)
(:domain AED)
(:objects SEFA - app Logback - lib...)
(:init
  (has-lib SEFA Logback)
  (= (version Jettision) 1004011000)...)
(:goal (and (ddos  SEFA)...))
```

All concrete typed objects in the system and environment

The initial state of the system and the environment

Attack goal

Figure. Example of PDDL problem file

Solutions for RQ1 (Modeling)

RQ1: Modeling the App's Endogenous System and its External Environment

How to efficiently model a microservice app's internal system and external environment within the Knowledge module?

- Update in a timely manner, consistently reflect changes and the system's latest runtime state.

Solution

- **Using PDDL 2.0 syntax** to **define** the system's internal configuration and external environment in the PDDL **"domain file" (AED)** and **"problem file" (AEDI).**
- The initial versions of AED and AEDI are specified in the offline stage and are pre-stored in the Knowledge module.
- At runtime, AED and AEDI are maintained by the Knowledge module. **Before planning APs** based on AED and AEDI to assess the risk of a specific implementation, the **Analyzer** must first **update AED and AEDI** using the latest observations collected by the Monitor.

AED & AEDI Specification

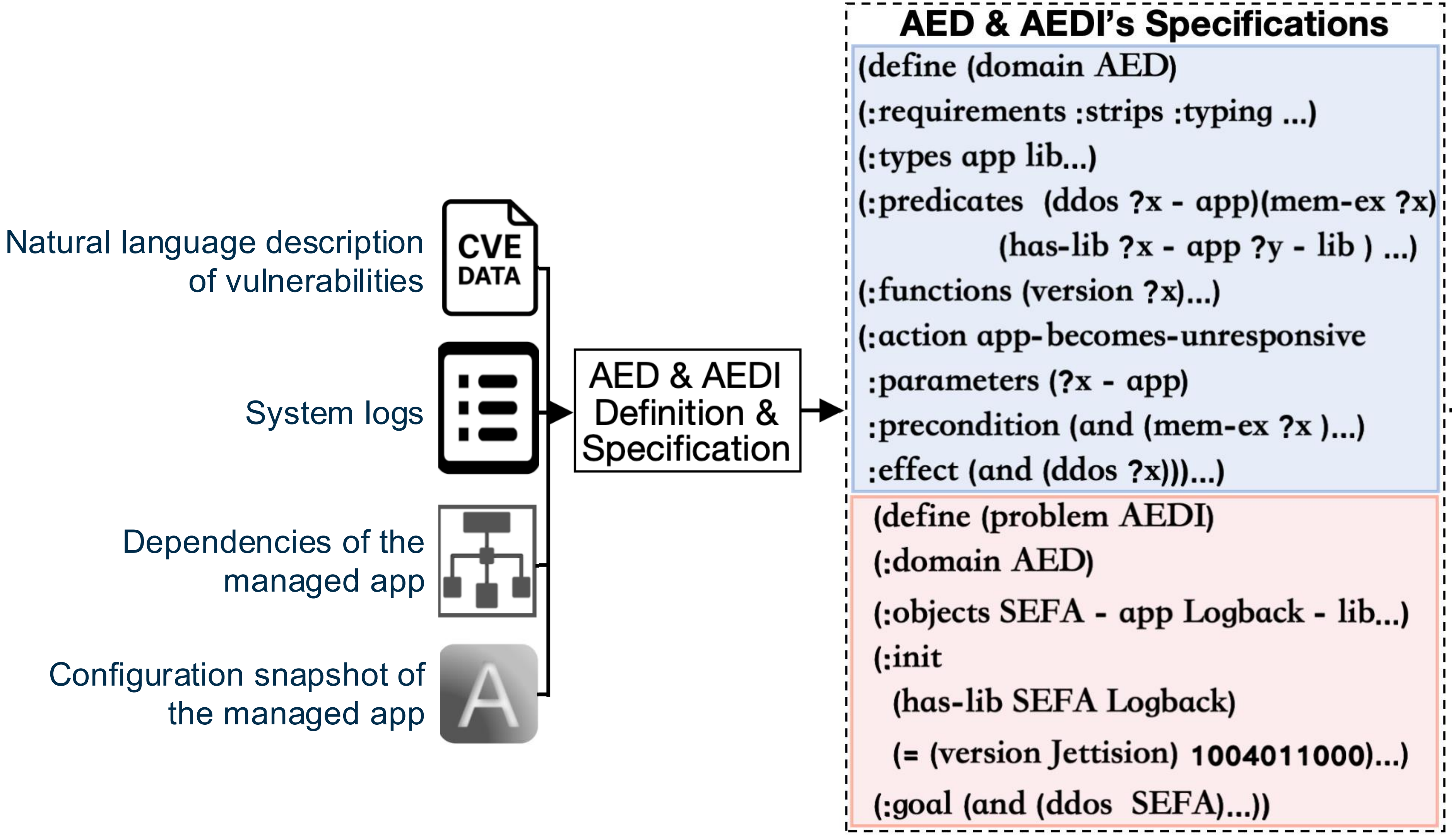


Figure. Workflow of specifying PDDL domain (AED) and problem files (AEDIs) to model the system configuration and environment

Abbreviation

AED: Adversarial Environment Domain;
AEDI: Adversarial Environment Domain Instance;

Contribution: Conceptual Framework of RAdaptSQ

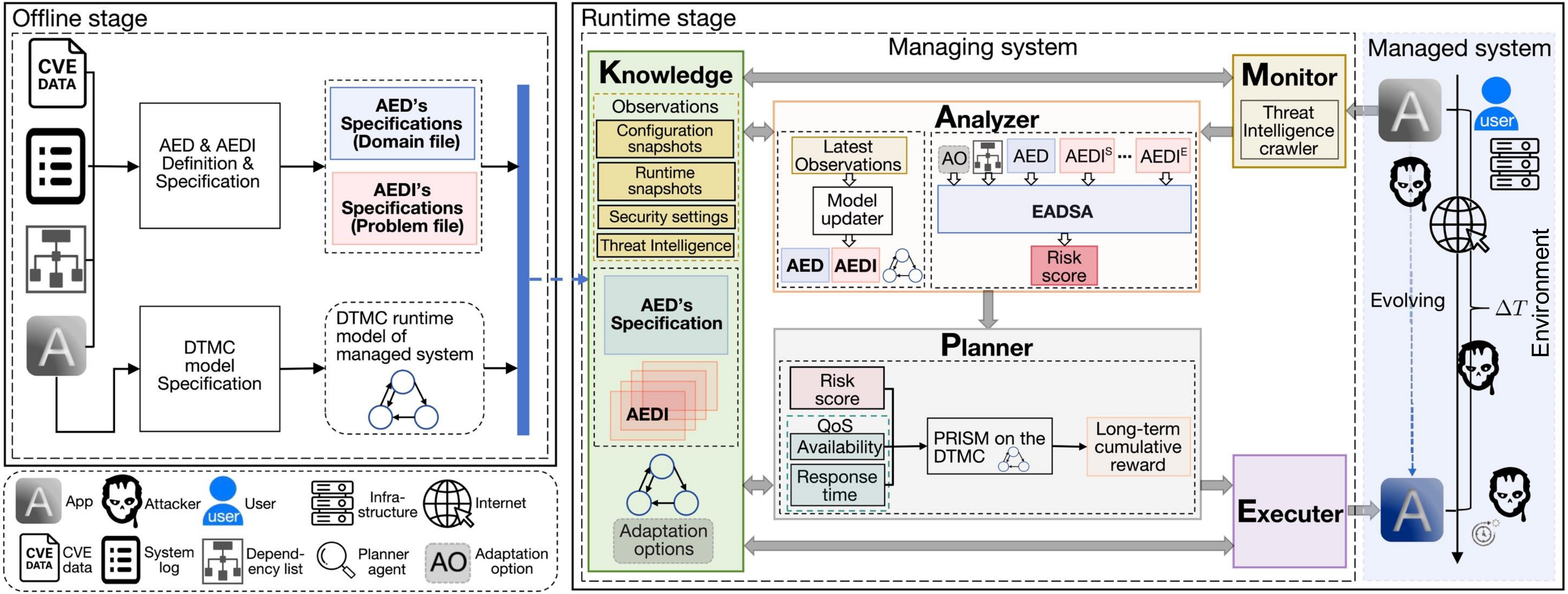


Figure. Conceptual framework of RAdaptSQ

Abbreviation

AED: Adversarial Environment Domain; AEDI: Adversarial Environment Domain Instance; EADSA: Environmental-Aware Dynamic Security Assessment

Contribution: Conceptual Framework of RAdaptSQ

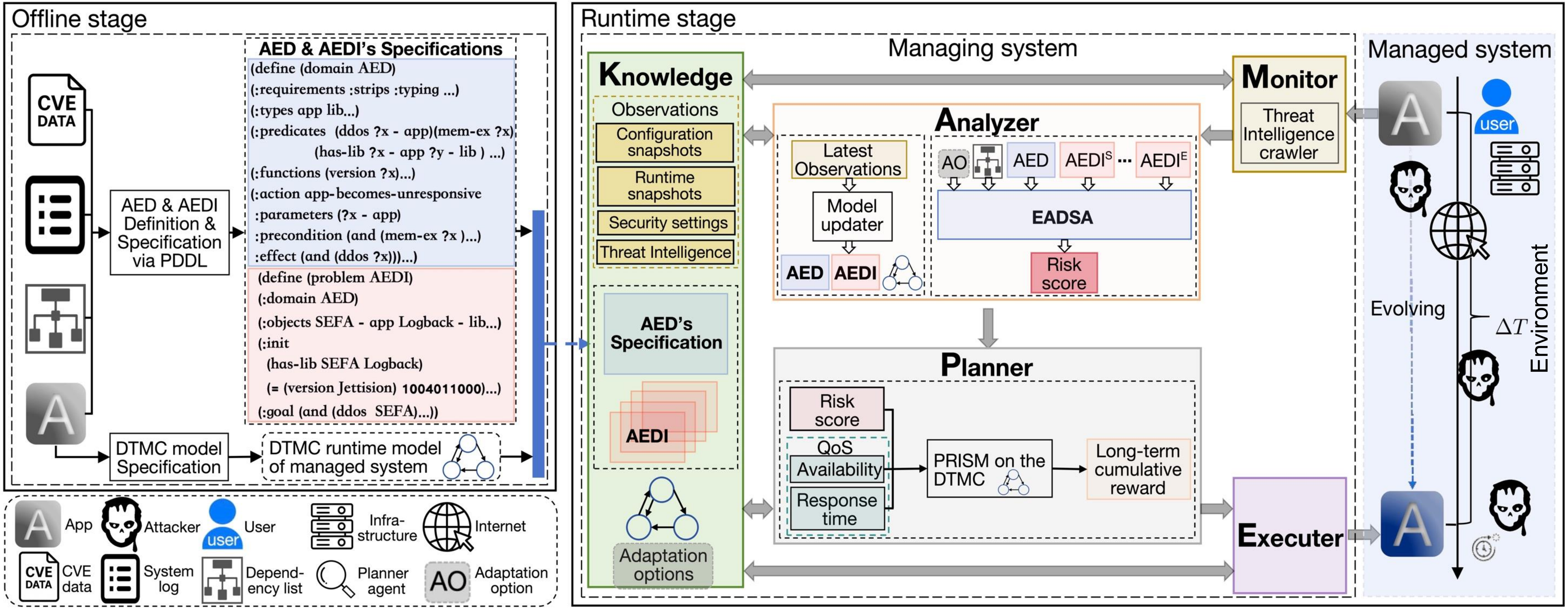


Figure. Conceptual framework of RAdaptSQ

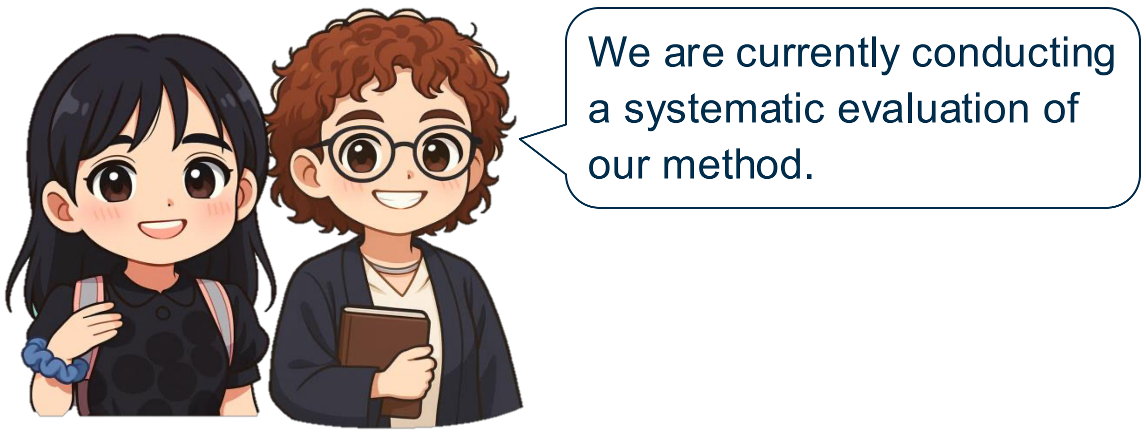
Abbreviation

AED: Adversarial Environment Domain; AEDI: Adversarial Environment Domain Instance; EADSA: Environmental-Aware Dynamic Security Assessment

Evaluation Plan

Research Questions

- RQ1: Effectiveness without runtime change
- RQ2: Effectiveness under runtime change
  (Simulate 3-5 types of runtime changes)
- RQ3: Scalability



Case study & Metric & Baseline

Case study:

- SEFA
- A classic microservice demo application (composed of 4-5 microservices) in the SAS community.

Metric:

- Runtime decision time: wall-clock time to
- Long-term cumulative reward: quantifies the expected long-term benefit of an adaptation.

Compared method:

- RAMSES (doesn’t consider security)
- AQUA (offline security assessment)
- RAdaptSQ’s ablation variants

Summary & Future Work

Summary

- Problem statement: Rapid attack surface fluctuation and security drift faced by microservice applications caused by their rapid changes
- Contribution: Propose the conceptual framework of RAdaptSQ, a real-time AI planning-driven and environment-aware self-adaptive system
- Objective: Automatically adapts the managed microservice application to changing attack surfaces, while continuously minimizing the security risk and ensuring acceptable QoS at runtime.

Future work

- RAdaptSQ's Monitor module periodically captures *CVE and limited behavioral statistics* from the Internet threat intelligence

- Large-scale, information-rich TI remains underutilized
- Develop an SAS-compliant mechanism for automated Threat Intelligence (TI) extraction and correlation analysis to better handle evolving attack surfaces

Thank you!

Questions?

Interested in more?
If you're interested,
please visit **sasis.kastel.kit.edu**

Email:
lin.cui@kit.edu
raffaela.mirandola@kit.edu

References

1. V. Riccio, G. Sorrentino, E. Zamponi, et al., “RAMSES: an artifact exemplar for engineering self-adaptive microservice applications,” in Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2024, pp. 161–167.
2. Hoffmann, J. (2003). The Metric-FF Planning System. Journal of Artificial Intelligence Research, 20, 291–341.
3. Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. JAIR, 20, 61–124.
4. Riccio, V., Sorrentino, G., Camilli, M., Mirandola, R., & Scandurra, P. (2023). Engineering Self-adaptive Microservice Applications. International Conference on Service-Oriented Computing, 227–242.
5. Weyns, D. (2020). An introduction to self-adaptive systems: A contemporary software engineering perspective. John Wiley & Sons.
6. Kwiatkowska, M., Norman, G., & Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. International Conference on Computer Aided Verification, 585–591.
7. Camilli, M., Luccioletti, F., Mirandola, R., & Scandurra, P. (2024). Integrated QoS- and Vulnerability-Driven Self-adaptation for Microservices Applications. ICSOC 2024, 55–71.
8. Jayalath, R. K., Ahmad, H., Goel, D., Syed, M. S., & Ullah, F. (2024). Microservice vulnerability analysis: A literature review. IEEE Access.
9. Gartner Peer Community. (2023). Microservices Architecture: Have Engineering Organizations Found Success? https://www.gartner.com/peer-community/oneminuteinsights/omi-microservices-architecture-have-engineering-organizations-found-success-u6b