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

Lin Cui
lin.cui@kit.edu
Karlsruhe Institute of Technology (KIT)

Raffaela Mirandola
raffaela.mirandola@kit.edu
Karlsruhe Institute of Technology (KIT)

Abstract

Microservice applications' dynamism expands the attack surface and causes security drift, rendering static security controls ineffective. Existing self-adaptive systems rarely tackle these dynamic security challenges. To address this, this paper provides the conceptual framework of RAdaptSQ, a real-time AI planning-driven self-adaptive system. It enhances the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) loop with a Planning Domain Definition Language (PDDL)-based adversarial domain and dynamic security assessment. It adapts microservices to evolving threats while preserving Quality of Service.

Index terms—MAPE-K loop, self-adaptive system, microservice application security

1 Introduction

Microservice-based Applications (Apps) improve modularity by decomposing monolithic functionality into independent, single-purpose services. This simplifies updating, scheduling, and scaling and allows integration of services from different providers. Consequently, microservices have been widely adopted across industries: a Gartner (2023) survey found that 74% of respondents already use a microservices architecture, while the remaining 23% plan to [7]. However, a single microservice may be updated independently 10–100 times per day [9]. The attack surface of microservice-based Apps can thus fluctuate rapidly, causing critical issues such as *security drift*, where static security controls with fixed strategies become outdated and ineffective shortly after deployment. It is of an urgent need to equip microservice-based applications with adaptable, long-lived security mechanisms that can continuously safeguard these Apps against changing attack surfaces and evolving threats.

Traditional static methods are obviously inadequate for handling such dynamic changes at runtime. For example, Palladio-based approaches provide powerful security modeling capabilities, but they are primarily focused on the design time [6]. Similarly, Attack Trees (ATs) [4] are widely used offline to model and assess system security statically. Even so, they lack support for partial updates during runtime, limiting their applicability in systems with rapidly changing runtime conditions. Self-adaptive systems (SASs) [5] can autonomously adjust to system and environment changes because they combine a managed system that provides core functionality and a managing system that monitors and adapts it. The managing system typically uses a MAPE-K feedback loop [5] to achieve adaptation goals with minimal human intervention. AQUA [8] is the first to make security enhancement an adaptation objective for microservices Apps by integrating dynamic security assessment into an adaptive MAPE-K loop. However, AQUA's security assessment is performed offline and remains static at runtime, so it cannot handle fluctuating attack surfaces. Therefore, to enable a MAPE-K loop that both enforces security thresholds while maintains acceptable QoS for Apps at runtime, there are two major challenges:

- **C1 (Modeling):** How to efficiently model a microservice app's internal system and external environment within the Knowledge module, so that the model can update in a timely manner to reflect changes and consistently represent the system's latest runtime state?
- **C2 (Dynamic assessment):** How can the MAPE-K loop dynamically assess the runtime security of the system based on the latest internal and external states while meeting the stringent time requirements of microservice environments?

In this paper, jointly addressing the above challenges, we present a conceptual framework of RAdaptSQ: a real-time AI-planning-driven and environment-aware self-adaptive system. Based on the observations collected by Monitor, RAdaptSQ maintains a Planning Domain Definition Language (PDDL)-specified Adversarial Environment Domain (AED) in the Knowledge module, which can be continuously updated through the MAPE-K loop during runtime to reflect the latest system and environment states (**C1**). Additionally, upon the most recent context, RAdaptSQ performs Environmental-Aware Dynamic Security Assessment (EADSA) during the analysis phase (**C2**). By continuously selecting and applying optimal adaptations, RAdaptSQ dynamically adapts the managed microservice app to evolving attack surfaces while ensuring acceptable QoS at runtime. The rest of this paper is organized as follows: Section 2 introduces the preliminaries; Section 3 presents RAdaptSQ's conceptual framework; Section 4 describes the evaluation plan; and Section 5 concludes.

2 Preliminaries

AI Planning and PDDL. AI planning, a key area of Artificial Intelligence (AI), focuses on automatically generating action sequences (plans) to solve planning problems

[2]. The PDDL standardizes problem description via two files: (a) The domain file, which defines the environment model by specifying types, predicates and actions. Predicates are Boolean relations describing state properties. Action templates include parameters, preconditions (conjunctions of predicates that must hold), and effects (predicate additions or deletions resulting from execution); (b) The problem file, which instantiates the domain for a specific scenario. It lists concrete objects, specifies the initial state as a set of grounded predicates, and goal states to be reached. A PDDL planner is the agent that receives these files as input and searches for an action sequence that transforms the initial state into one satisfying the goal.

Discrete-Time Markov Chains. Discrete-Time Markov Chains (DTMCs) [1] model systems with probabilistic state transitions over discrete time steps. A DTMC is a tuple $\mathcal{M}=(S,s_0,P,L)$, where S is a finite set of states, $s_0 \in S$ the initial state, $P: S \times S \to [0,1]$ the transition probability matrix with $\sum_{s' \in S} P(s,s') = 1$ for all $s \in S$, and $L: S \to 2^{AP}$ labels states with subsets of atomic propositions AP. For a microservice application, states S include: (a) entry points (s_0), (b) microservices, and (c) terminal outcomes, while transitions P represent API-call dependencies among the microservices. To capture non-functional properties such as availability, response time, and security, DTMCs are typically extended with a reward structure $r=(r_s,r_a)$, where $r_s : S \to R_{\geq 0}$ and $r_a : S \times S \to R_{\geq 0}$ assign non-negative rewards to states and transitions, respectively. In practice, DTMC models are built manually or derived from execution traces, with initial transition probabilities based on benchmark availability and updated online via the MAPE-K loop [8].

3 Conceptual Framework

This section outlines RAdaptSQ's conceptual framework (Figure 1), which comprises two main stages:

S1: Offline stage (C1& C2). In this stage, we analyze the managed application, and define and specify the models of the managed application and its environment in the Knowledge module before deployment. Specifically, we define the managed application's endogenous configuration and exogenous environment as the AED and its Adversarial Environment Domain Instance (AEDI) by taking advantages of AI planning. The AED defines the general schema encompassing entity types, states, relationships, numeric attributes and atomic actions. The AEDI instantiates these schema elements with concrete objects and values to capture the application's state at a specific time point and specifies goal states for security analysis. Using PDDL 2.1 syntax [2], the initial versions of the AED and AEDI are specified in the domain and problem files, respectively, with the initial settings and values. Figure 1 shows a concise depiction of AED and AEDI. Additionally, by following AQUA [8], we specify a DTMC runtime model to represent the managed application's probabilistic behavior during execution.

S2: Runtime management. In this stage, the enhanced MAPE-K feedback loop continuously optimizes the managed application by selecting and applying optimal adaptation options to ensure security and maintain acceptable QoS. Specifically, RAdaptSQ's MAPE-K self-adaptive loop operates as follows:

- **K (C1):** To reflect the highly dynamic microservice application and its external evolving environment, RAdaptSQ's Knowledge module maintains the AED and its AEDI with the latest observations. After the initial version of AED and ADEI are specified and stored in the Knowledge module offline, they are periodically and partially updated at runtime according to the fluctuations of both endogenous system and exogenous environment detected by the Monitor module. In addition, the DTMC model is also updated with the latest observation.
- **M (C1 & C2):** To support these updates, RAdaptSQ's Monitor not only periodically and asynchronously collects (a) configuration snapshots, (b) runtime snapshots, and (c) security settings, but also integrates a web crawler to continuously extract and pipe required relevant Threat Intelligence (TI) from the Internet, such as Common Vulnerabilities and Exposures (CVE) data and behavioral statistics in threat scenarios, into the Knowledge module.
- **A (C2):** RAdaptSQ's Analyzer is responsible for spotting the changes from the latest observations collected by the Monitor and partially updating the corresponding specifications in the domain or problem files accordingly. To enable runtime self-adaptation, upon having the updated AED and AEDI, RAdaptSQ's Analyzer is specifically embedded with an AI planning-driven EASDA method to dynamically prepare security risk scores for available adaptation candidates in near real-time. Specifically, to assess the security risk of available options, we integrate the strengths of Attack Path (AP) models [4] for attack vector visualization and risk assessment with the capabilities of AI planners for adaptive, real-time attack path planning, formulating the AP generation as an *Environment-Aware Attack Path Planning Problem* (p_{AP}) based on AED and its AEDI, where each AP is a sequence composed of step-wise atomic actions that transform from the system ground state at a given time into the designated compromised state. By specifying six p_{AP} instances, each targeting a goal state corresponding to one of the STRIDE categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege, six independent PDDL AI planners solve these problems and generate six risk-maximizing APs in a parallel way. The final security risk is quantified as a weighted sum of the six individual risk scores. Each score on one AP is calculated as the product of: (a) the product of per-step success probabilities along the AP and (b) the critical impact of the vulnerability exploited in the AP. Upon preparing the available implementation options along with their associated security risk scores and QoS metrics, RAdaptSQ's Analyzer conveys this information to the Planner via the Knowledge module.

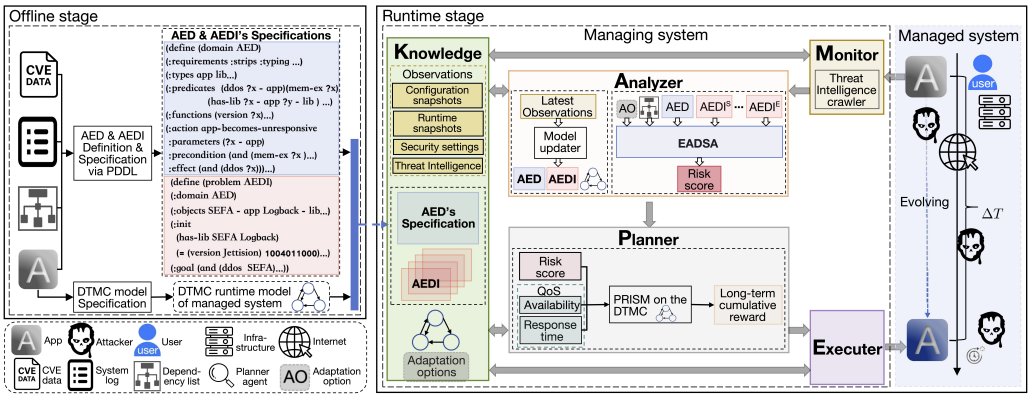

Figure 1: The conceptual framework of RAdaptSQ.

- **P:** Upon invocation, RAdaptSQ's Planner retrieves the available implementation options with their associated metrics from the Knowledge module. It then selects the optimal option by simulating the execution of managed microservice application over a finite horizon by running its DTMC model on PRISM [3] probabilistic model checker. A long-term reward is accumulated to quantitatively evaluate and compare the effectiveness of each option. Once the decision is made, the **Executor** is triggered to apply the selected option.

4 Evaluation Plan

To comprehensively verify RAdaptSQ's performance, we design three research questions: **RQ1 (Effectiveness without runtime change)**, **RQ2 (Effectiveness under runtimes change)**, **RQ3 (Scalability)**. The experiments will be conducted on two case applications, each comprising 4–6 microservices. Primary metrics are average wall-clock time and cumulative long-term reward (quantifies the expected long-term benefit of an adaptation), measured from adaptation onset to the Planner's decision, averaged over 10 identical runs. For **RQ1**, we compare RAdaptSQ with AQUA prior baselines [8], plus ablation variants of RAdaptSQ, in both cases. Each case comprises 5–6 scenarios varying: (1) configuration of the managed system (deployed microservices and available instances); (2) QoS level for each instance (availability, response time); and (c) vulnerability list per service implementation. For **RQ2**, we compare RAdaptSQ and AQUA in both cases under 3–5 runtime changes in configuration or environment to show RAdaptSQ's advantage in dynamic risk scoring versus AQUA's static offline approach. **RQ3** reuses RQ1's baselines on case one. We respectively measure offline (definition & specification) and runtime (AP planning & dynamic assessing & adaptation decision making) wall-clock time costs of all baselines by varying the vulnerability number of each service implementation and composed microservice number of an App. All experiments replicate identical operating conditions and analyse adaptation decisions at the plan component level.

5 Conclusion

To tackle the changing attack surface and evolving threat faced by dynamic microservice-based applications, this paper presents a conceptual framework of RAdaptSQ: a real-time AI planning-driven and environment-aware self-adaptive system. RAdaptSQ automatically adapts the managed microservice application to changing attack surfaces, while continuously optimizing their security and ensuring acceptable QoS at runtime. Although RAdaptSQ's Monitor module can capture CVE and limited behavioral statics from the Internet threat intelligence, large-scale and information-rich TI remains underutilized. Future work aims to develop an SAS-compliant mechanism for automated TI extraction and correlation analysis to better handle evolving attack surfaces.

References

[1] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[2] M. Fox and D. Long. "PDDL2. 1: An extension to PDDL for expressing temporal planning domains". In: *Journal of artificial intelligence research* 20 (2003), pp. 61–124.

[3] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of probabilistic real-time systems". In: *International conference on computer aided verification*. Springer. 2011, pp. 585–591.

[4] A. Shostack. *Threat modeling: Designing for security*. John wiley & sons, 2014.

[5] D. Weyns. *An introduction to self-adaptive systems: A contemporary software engineering perspective*. John Wiley & Sons, 2020.

[6] S. Seifermann et al. "Identifying confidentiality violations in architectural design using palladio". In: *ECSA-C202021* 2978 (2021).

[7] Gartner Peer Community. *Microservices Architecture: Have Engineering Organizations Found Success?* 2023.

[8] M. Camilli et al. "Integrated QoS-and Vulnerability-Driven Self-adaptation for Microservices Applications". In: *ICSOC 2024*. Springer, pp. 55–71.

[9] V. Heorhiadi et al. "Gremlin: Systematic resilience testing of microservices". In: *ICDCS 2016*. IEEE, pp. 57–66.