PREPRINT – accepted for publication at the
19th European Conference on Software Architecture (ECSA) 2025

SAFER-D: A Self-Adaptive Security Framework for Distributed Computing Architectures

Marco Stadler (Johannes Kepler University Linz, LIT Secure and Correct Systems Lab / Institute of Business Informatics – Software Engineering; marco.stadler@jku.at)

Michael Vierhauser (University of Innsbruck, Department of Computer Science; michael.vierhauser@uibk.ac.at)

Michael Riegler (ENGEL Austria GmbH, Information Security; michael.riegler@engel.at)

Daniel Waghubinger (Johannes Kepler University Linz; N/A)

Johannes Sametinger (Johannes Kepler University Linz, LIT Secure and Correct Systems Lab / Institute of Business Informatics – Software Engineering; johannes.sametinger@jku.at)

DOI: TBA

ABSTRACT

The rise of the Internet of Things and Cyber-Physical Systems has introduced new challenges on ensuring secure and robust communication. The growing number of connected devices increases network complexity, leading to higher latency and traffic. Distributed computing architectures (DCAs) have gained prominence to address these issues. This shift has significantly expanded the attack surface, requiring additional security measures to protect all components – from sensors and actuators to edge nodes and central servers. Recent incidents highlight the difficulty of this task: Cyberattacks, like distributed denial of service attacks, continue to pose severe threats and cause substantial damage.

Implementing a holistic defense mechanism remains an open challenge, particularly against attacks that demand both enhanced resilience and rapid response. Addressing this gap requires innovative solutions to enhance the security of DCAs.

In this work, we present our holistic self-adaptive security framework which combines different adaptation strategies to create comprehensive and efficient defense mechanisms. We describe how to incorporate the framework into a real-world use case scenario and further evaluate its applicability and efficiency. Our evaluation yields promising results, indicating great potential to further extend the research on our framework.

1 Introduction

The hype surrounding the Internet of Things (IoT) and Cyber-Physical Systems (CPSs) drives a surge in Internet-connected devices. Managing this many devices requires careful organization, often achieved through diverse architectural styles and patterns [9]. Instead of a centralized infrastructure, distributed computing architectures (DCAs) are used to provide reduced latency, real-time analysis, high scalability, low operational cost, and improved quality of service [9]. Although distributed computing helps operators deal with complexity, it affects the attack surface of these architectures. The sheer number of devices and their convolution, heterogeneity, diversity, interoperability, portability, mobility, location, topology, and distribution of objects cause an increase in attack surface and make the architecture susceptible to cyberattacks (details in [26]). In particular, interoperability and interdependency are crucial factors in this context [22]. A failure caused by an attacker to one subsystem can lead to cascading failures, rendering the whole DCA inoperable [4]. Particularly on critical infrastructure, a successful cyberattack can cause severe harm [29, 24, 6], ranging from compromised databases to human injury. Recent incidents, such as the record-breaking 5.6 Tbps Distributed Denial of Service (DDoS) attack targeting Cloudflare's infrastructure [35], highlight the tangible risks faced by the industry. Studies [29, 24, 6] demonstrate that the scientific community recognizes these threats, emphasizing the urgent need for resilient security strategies in DCAs.

The reasons for such incidents still occurring are manifold. Security measures often only provide isolated and passive defense mechanisms, severely limiting their effectiveness [34]. Passive mechanisms usually rely on predefined rules. For example, they may block network packets based on known signatures [23]. They follow a "detect then patch" philosophy, meaning they are only effective after an attack. As a result, they cannot adapt proactively and respond to threats in real-time [34]. – As a result, the question of securing DCAs with a holistic and active security solution to efficiently adapt to the evolving threat landscape remains. In this paper, we propose a novel idea of combining self-adaptive architectural patterns to improve the security of DCAs. More specifically, we leverage, among others, hierarchical adaptation strategies [33], adaptation strategies used in Systems of Systems (SoS) [32], and the concept of security levels (cf. Section 3.2) to enhance the overall resilience of DCAs in the event of attacks. To the best of our knowledge, no prior work combines hierarchical, collaborative, and decentralized adaptation strategies to ensure self-adaptive security under partial system failure. Our framework addresses this gap through its dual-loop architecture and adaptation modes. As part of this, we claim the following contributions:

- Novel, Secure Adaptive Framework for Efficient Resilience in Distributed computing architectures (SAFER-D) that allows for security adaptations at the architectural level, even when under attack.
- Prototypical implementations using real-world edge computing

architectures for component reuse.

- Evaluation of applicability and efficiency of SAFER-D, based on realistic security use case scenarios.

2 Motivating Architectural Challenges

In an increasingly interconnected world, the ability to dynamically adapt to emerging threat scenarios is becoming ever more critical. Adaptive threat monitoring focuses on continuously observing systems for unusual or suspicious activities and adjusting responses based on evolving contexts. This adaptability is vital in defending against cyberattacks targeting CPSs/IoT systems where static approaches are insufficient. Intrusion Detection Systems (IDS) [19] often leverage an adaptive approach to detect and withstand cyberattacks. An IDS commonly monitors network or system activities, detects potential security threats, and executes appropriate countermeasures or sends alerts. – This process largely aligns with the MAPE-K (Monitor, Analyze, Plan, Execute on a shared Knowledge base) loop [16], a foundational pattern for self-adaptive system architectures.

Consider the following DCA example, which will serve as our running case: Edge computing, combined with fog and cloud computing, places substantial computing and storage resources at the (physical) outer "edges," where data is generated. The system processes data directly, forwards only aggregated data to fog computing components, aggregates it again, and then sends only the relevant data to the next central server, continuing this pattern[5]. An IDS applied to one of the edge devices of such an edge computing architecture *monitors*, for instance, the network traffic on the device, *analyzes* the data to identify anomalies or suspicious patterns (e.g., a DDoS flooding attack), *plans* appropriate responses (e.g., block a specific Internet Protocol (IP) address), and *executes* the countermeasures (append IPs to a blocklist) or generates alerts, all supported by a *knowledge* base to enhance detection accuracy and adaptability. In the following, we highlight **Architectural Challenges (ACs)** concerning a self-adaptive security framework for DCAs. These challenges are informed by our industry collaborations and supported by academic literature, based on recurring needs identified in regular technical meetings and through a structured review of recent research on self-protective and adaptive systems.

AC 1 - Managing adaptation in complex and large-scale DCAs using a single MAPE-K loop is insufficient: Weyns *et al.* formalized a series of architectural patterns comprising multiple interconnected MAPE-K loops to deal with large, complex, and heterogeneous systems [33]. Among others, they introduced the hierarchical control pattern. This pattern manages the complexity of self-adaptation by establishing a layered separation of concerns through a hierarchy of MAPE-K loops. Loops at lower layers operate on a short time scale, ensuring that the portion of the system under their direct control adapt promptly. Higher levels operate on a more global/strategic scale over an extended period. Ultimately, the MAPE-K loop at the system's summit determines the system's overarching adaptation objectives. Applying the hierarchical MAPE-K pattern to our aforementioned example implies that, for instance, the fog nodes in the edge computing architecture use the monitoring data of multiple underlying edge devices for the adaptation loops and then roll out a collective adaptation strategy for all devices associated with the respective fog node.

AC 2 - Hierarchical adaptation strategies break when intermediate nodes are compromised: In our example, the edge computing architecture follows a hierarchical organization, which creates dependencies. Suppose fog node in the hierarchy is compromised and unavailable, e.g., due to a successful attack. In that case, the underlying edge devices will not receive adaptation updates and thus remain susceptible to subsequent cyberattacks. Regarding security, individual system components must be independent from an operational and managerial viewpoint, exhibiting SoS characteristics [32, 7]. Self-adaptive architectural patterns for SoS have been frequently studied [32]. One of them, the *Collaborative Adaptations* style, allows for adaptations on a control-theoretic level. Additionally, the architecture allows for interactions among the managed systems, supporting collaboration between the subsystems. Using this pattern, the SoS can adapt comprehensively while considering each component [32]. Subsystems can then coordinate locally and continue adaptation without relying on the compromised node.

AC 3 - Security mechanisms must remain effective even when parts of the system are already compromised: The Risk of an attacker compromising a system is defined as follows [15]:

$$R = \{s_i, p_i, x_i\}, \qquad i = 1, 2, ..., N \tag{1}$$

where R represents the risk; s an undesirable event scenario description; p the probability of the scenario; x the potential damage caused by the scenario; and N the number of possible scenarios that may cause damage to a system. It is important to note that p and x are not constants but rather evolve over the time of an attack, serving here as a conceptual model to illustrate this dynamic. In our edge computing example, if an attacker compromises one edge device, the probability that they successfully compromise another edge device increases (i.e., p in Eq. (1) evolves). If attackers can circumvent security mechanisms once, they can reproduce the attack on other devices with the same security mechanisms. Similarly, once a publicly available proof of concept exploit exists for a known vulnerability, the probability of reproduction increases. A compromised single system can also influence the "neighbors" communicating with the device. Depending on the type of attack, the attackers move laterally [12] and propagate [1] to the controlling (managing) systems [17], i.e., in the best case (from an attacker's perspective), up the hierarchy to attain more and more control. Security solutions are required to respond promptly to attacks and must be able to cope with already compromised system components.

AC 4 - Security adaptations must consider the evolving criticality and impact of threats over time: Not only does the probability evolve, but also the potential damage (cf. x in Eq. (1)). For instance, if the running example's edge computing system supports autonomous driving, it must prioritize safety to protect human lives [20]. A collision becomes more likely if the vehicle malfunctions (due to an attack). Autonomous vehicles must slow

down, or even shut down entirely, after detecting malicious activity. To cope with these evolving factors, security mechanisms must support means of criticality.

3 The SAFER-D Framework

In the following, our novel SAFER-D framework addresses the architectural challenges (**AC 1-4**).

3.1 Core Components

Fig. 1 depicts a high-level overview of SAFER-D. The core idea is that SAFER-D is deployed on every single subsystem of the DCA. In the running case, this implies server, fog, and edge computing subsystems each run an instance of SAFER-D. Naturally, these instances must be tailored to the respective hardware capabilities, meaning that resource-intensive tools may only be available on more powerful subsystems, while lightweight variants are deployed on constrained edge devices. `Subsystem n` represents one of N subsystems in the computing architecture, e.g., a single edge device, where the SAFER-D instance communicates with other SAFER-D instances deployed on the rest of the architecture (i.e., the `Subsystems of Interest`). With the term "subsystem," we refer to a single independent computing node. As part of SAFER-D, we use two types of MAPE-K loops: First, `Local MAPE-K` represents the "traditional" adaptation loop commonly found in a self-adaptive system, running locally and only internally on each subsystem. Second, the adaptation loop `Global MAPE-K` has a dedicated communication channel to other subsystems in the architecture, i.e., other edge and fog devices running a SAFER-D instance.

`Local MAPE-K`: The `Local Runtime Monitor` gathers data from the `Managed system` and forwards it to the `Local Adaptation Middleware`. The `Local SL Manager` (SL: Security Level) plans an appropriate security level (cf. Section 3.2). Finally, a `Local Execution Adapter` executes the planned adaptations on the `Managed system` (M ⇒ A ⇒ P ⇒ E).

`Global MAPE-K`: This adaptation loop serves two purposes: (*i*) The loop allows for holistic adaptations together with other subsystems (cf. **AC 1**) and (*ii*) ensures a prompt response in the event of an attack (cf. **AC 3**).

(*i*) The `Global SL Monitor` forwards adaptation information from the other `Subsystems of Interest`. In our running case for a fog subsystem, the subsystems of interest are the superordinate server subsystems and the subordinate edge subsystems. The adaptations, i.e., the security levels of other computing systems, are then forwarded to the `Local Adaptation Middleware`. The `Local MAPE-K` can then, besides the local monitoring data, also take comprehensive information from other connected systems into account for performing adaptations (M/M ⇒ A ⇒ P ⇒ E).

(*ii*) Security incidents can lead to isolated subsystems, disrupting their integration into the `Subsystems of Interest` (cf. SoS in **AC 2**). When this happens, it is critical to ensure prompt adaptation times to prevent other subsystems from being compromised. Disruptions can cause delays in the adaptation time, e.g., due to unanswered requests during an attack (cf. **AC 3**). The `Global MAPE-K` takes these disruptions into account by adapting the `Managed API` (Application Programming Interface). The API configuration uses two operational modes: Full Adaptation (FA) when all other subsystems are available and Partial Adaptation (PA) when at least one subsystem of interest is not reachable. In both modes, the `Global MAPE-K` will be executed, and the `Managed API` adapts continuously. The difference lies in the number of `Subsystems of Interest` checked for adaptation updates (cf. details in Section 3.3). The switch between these modes is decided within the `Global MAPE-K`: The `Managed API` gathers the *Global SL Adaptations* and forwards the *Network Status Data* to the `Global Network Monitor`. The `Global Adaptation Middleware` checks the connections to other subsystems and detects if a subsystem in the `Subsystems of Interest` is unresponsive. The `Global Config Manager` then triggers the respective mode, and the `Global Execution Adapter` carries out the adaptation by reconfiguring the `Managed API` (M ⇒ A ⇒ P ⇒ E).

3.2 Security Levels

Security (criticality) levels (SLs) in SAFER-D represent varying degrees of protective measures specifically tailored to the current risk and criticality of a system (cf. **AC 4**). The concept is inspired by so-called "readiness levels," found, for instance, in the US military's DEFense readiness CONdition (DEFCON) stages [30]. They enable dynamic adaptation of security mechanisms, ensuring system can escalate or de-escalate their defenses based on evolving threats or environmental conditions. We introduce SLs as part of SAFER-D's `Local SL Manager` to adjust security based on dynamic changes (cf. x and/or p in Eq. (1)). For example, in our running case, under normal conditions, an edge device operates at DEFCON 5. If unusual behavior suggests a potential attack, it escalates to DEFCON 4, increasing the monitoring of system load, as sudden spikes may indicate a DDoS attack. The DEFCON level continues to adapt as risks rise. The highest escalation is DEFCON 1 in critical cases (e.g., a confirmed DDoS attack) where the system may even shut down. The discrete SL design encourages security engineers to define clear, scenario-specific countermeasures per threshold, though we acknowledge this limits flexibility in multi-threat situations; addressing such conflicts will be a focus in future iterations of the framework.

The `Local Adaptation Middleware`, and the subsequent `Local SL Manager`, consider two sources when deciding on the appropriate SL. The local *Security Events* provided by the `Local Runtime Monitor` (M) and the *Global SL Adaptations* of the other `Subsystems of Interest` in the architecture forwarded by the `Global SL Monitor` (M). SAFER-D supports a human-in-the-loop approach for returning to less restrictive security levels.

3.3 Full and Partial Adaptation Mode

The `Global MAPE-K` leverages two distinct operational modes to speed up the global adaptation loop (cf. Section 3.1): *Full Adaptation (FA)* and *Partial Adaptation (PA)*. In both modes, SAFER-D tries to gather adaptation information from other `Subsystems of Interest`.

In a perfect world, a single subsystem can always communicate with the other subsystems of interest. Consider the example in Fig. 2a, showing a conceptual edge computing architecture with server, fog, and edge computing subsystems. For instance, `#2`

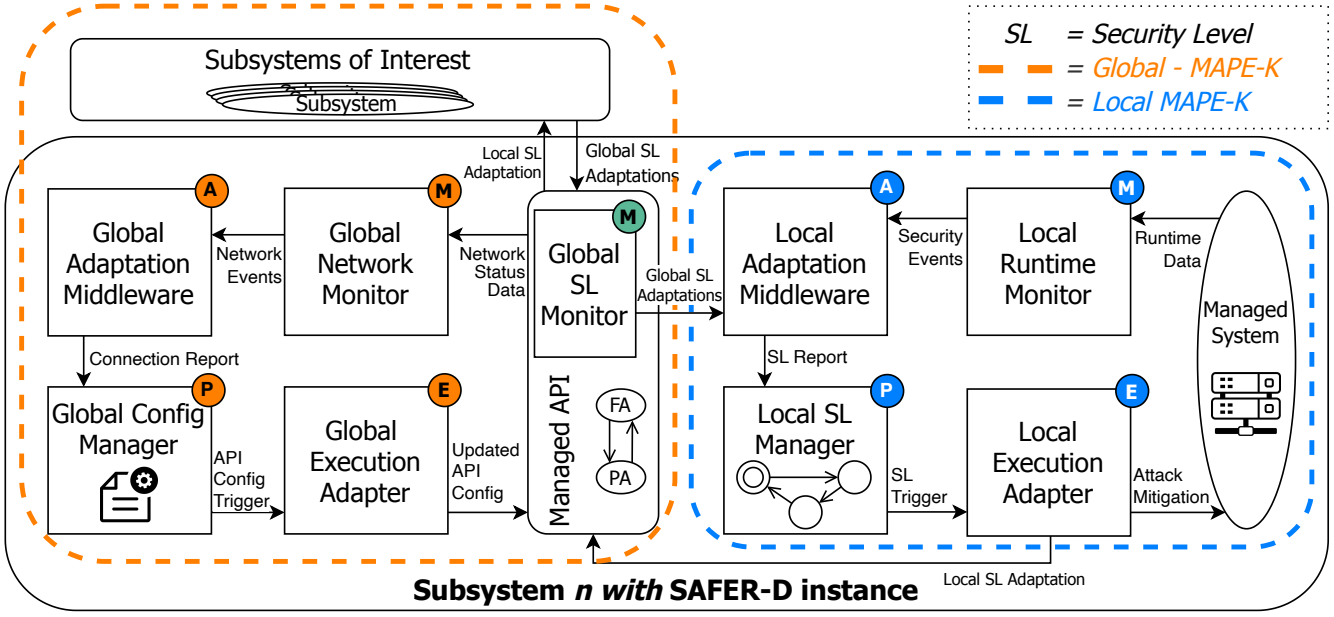


Figure 1: High-level Overview of SAFER-D.

communicates updates to its superordinate server subsystem #1 and its subordinate subsystems #6 and #4. During FA, full adaptation within the DCA is possible, and every subsystem can receive respective adaptation updates from every other subsystem. For the example in Fig. 2a, the subsystems adapting and exchanging data consists of all subsystems: [#1, #2, #3, #4, #6, #7, #5, #8].

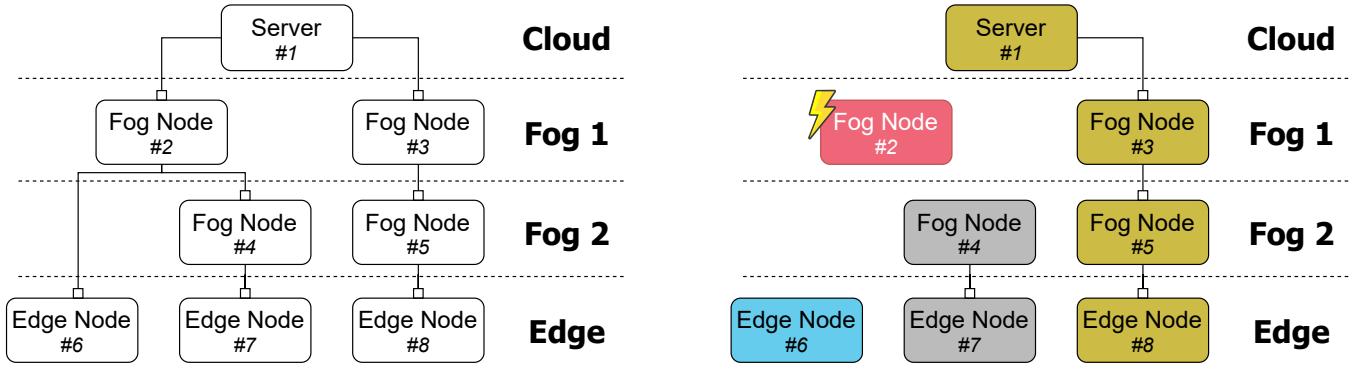
In practice, a subsystem of interest can become unresponsive or unavailable due to an attack. In such a case, fast and timely adaptation becomes one of the most important properties for preventing and potentially repelling attacks on other uncompromising subsystems. The unavailable subsystem interferes with this goal. For instance, when subsystem *A* sends a request to the unresponsive subsystem *B*. *A* cannot continue the adaptation until it reaches a timeout, thereby delaying the adaptation loop. The problem is exacerbated when more than one subsystem is already compromised. – In such a case, SAFER-D uses the PA mode, displayed in Fig. 2b. In the example, subsystem #2 is unresponsive due to an attack. As part of SAFER-D's `Global MAPE-K` loop, the remaining (still available) subsystems form adaptation subgroups based on the availability of connections. In the example in Fig. 2b, the architecture is split into three adaptation subgroups: [#1, #3, #5, #8], [#4, #7], and [#6]. Splitting the architecture and excluding the unresponsive subsystem #2 helps maintain adequate adaptation times. The splitting (i.e., the mode switch) in SAFER-D is performed by reconfiguring the `Managed API` and dictating which subsystems of the `Subsystems of Interest` should be requested for adaptation updates (i.e., contribute to the *Global SL Adaptations*) and which are ignored. As part of the `Global MAPE-K` loop, a recovery strategy is in place to check the responsiveness of unavailable subsystems, similar to a heartbeat function. Once a subsystem is up and running again, the subsystem is reintegrated into the adaptation set until the system can switch back to the FA mode.

In the following, we further describe the interplay of the two MAPE-K loops. SAFER-D is efficiently greedy in the sense of how it conducts adaptations. The more subsystems are available, the more information can be used. SAFER-D always tries to connect to the other `Subsystems of Interest` as part of the `Global MAPE-K`. When this is impossible, it still uses its PA mode to receive as many timely adaptations as possible. In the worst case, no other subsystems are available (i.e., (M) is exhausted). In this case, SAFER-D can at least adapt locally (i.e., (M) is the only source for adaptations). The interplay of the loops is also visualized in Fig. 3. The novelty of SAFER-D lies in its flexibility in different situations. SAFER-D is capable of dealing with interruptions and can, whenever necessary, adapt so that two MAPE-K loops are always running efficiently: One to adapt the `Managed System`, and one to adapt the `Managed API`. This flexibility directly adheres to the resilience of the architecture SAFER-D is deployed to and marks the main contribution of our framework.

4 Evaluation

We executed a series of rigorous experiments. We evaluated the *Feasibility* and *Security* in an earlier framework version [28] here, we focus on *Applicability* and *Efficiency*. to evaluate our framework's general applicability and efficiency. In this section, we first describe the research questions, use case scenarios, evaluation setup, results, and finally, the answers to our research questions.

RQ1: *(Applicability) To what extent can SAFER-D be applied to execute security adaptations, and what is the integration effort when embedding it into existing system architectures?* With this first research question, we aim to qualitatively evaluate the applica-



(a) Architecture during *FA*.

(b) Successful attack at subsystem #2 and resulting *PA* adaptation subgroups.

Figure 2: Edge computing architecture in the use case scenario.

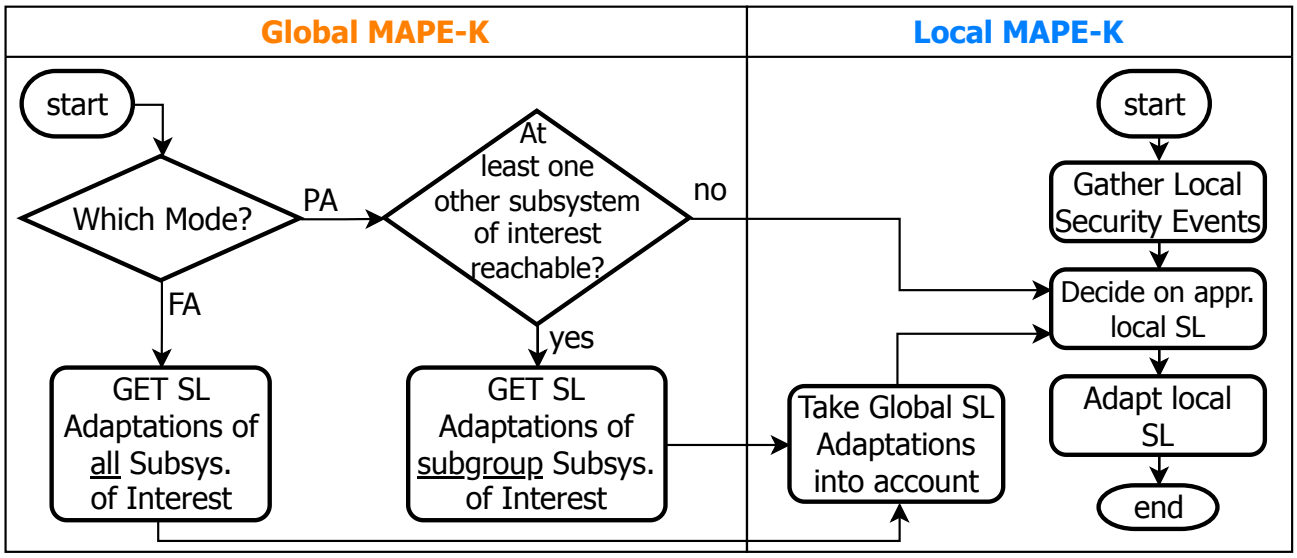


Figure 3: Simplified flowchart illustrating `Global MAPE-K`'s influence on the `Local MAPE-K`.

bility of SAFER-D. Applicability is essential to bridge theory and practice, and we try to approach an answer by addressing some of the key factors used in similar evaluation setups (see [18]). For measuring the integration effort, we report on the reusability of components and the required resources to use SAFER-D in a given scenario, i.e., the time and Lines of Code (LoC) it takes to configure and run SAFER-D.

RQ2: *(Efficiency) To what extent can SAFER-D be used to execute security adaptations efficiently?* With the second research question, we quantitatively determine the efficiency of SAFER-D. Since a timely adaptation is paramount during an attack (cf. [8] and **AC 3**), short adaptation times are a crucial indicator of SAFER-D's efficiency. We assess the *Time to Adapt* (TtA) on the deployed architecture of all MAPE-K loops. First, we measure the *Time to Adapt for Security Levels* (TtA_{SL}) to evaluate the adaptation times of (M)/(M) ⇒ (A) ⇒ (P) ⇒ (E). Hence, the adaptations from one SL to another. Second, we measure the *Time to Adapt for Global Adaptations* (TtA_G) to measure the switch between PA and FA to form adaptation subgroups. Hence, TtA_G reflects the time it takes for a (M) ⇒ (A) ⇒ (P) ⇒ (E) loop configuration to take effect on the `Managed API`.

4.1 Use Case Scenario

Our use case is motivated by a real-world example provided by our industry partner ENGEL Austria GmbH. The company is a large machine manufacturing enterprise operating in over 80 countries worldwide, with several thousand employees. The company is one of the leading manufacturers of industrial injection molding machines. With its globalized status, the company provides multiple large edge computing architectures distributed worldwide for its customers. Machine data (e.g., production cycles, usage data) is collected and distributed to central cloud computing nodes via a

hierarchically structured edge computing architecture. This data is used, among other things, for predictive maintenance, remote support, and data analysis to increase production efficiency and reduce energy consumption and scrap. Multiple machines are connected via edge devices, which are again connected to fog computing nodes and ultimately report to a central cloud server. Although the machines can be operated standalone, downtime of the edge computing architecture due to a security incident or a cyberattack can impede normal operation and cause financial losses. With its underlying edge computing architecture and security requirements, the company provides an excellent example for evaluating the different aspects of SAFER-D. For the evaluation, we focused on DDoS attacks on the edge computing architecture [34]. More specifically, we used SAFER-D to adapt in the event of an Internet Control Message Protocol (ICMP) flood attack. In such an attack, an excessive number of ping requests are sent to the target, thereby clogging up its resources, such as network bandwidth and processing capabilities. The use case's goal is to dictate security-level adaptations among the edge computing subsystems during an ICMP flood attack. If a system becomes unavailable due to the attack, the respective subsystems shall switch into the PA mode to allow for respective adaptations within the remaining adaptation subgroup.

4.2 Evaluation Setup

Based on the conceptual framework above, we created a prototype implementation in Python to conduct the evaluation. In the following, we provide a brief overview of SAFER-D's instantiation (see details in suppl. material). The industry partner provided us with edge devices (hardware) utilized on-site at their manufacturing plants. We use these devices to replicate a typical edge computing architecture for the evaluation. The evaluated architecture is depicted in Fig. 2a and consists of four layers: one cloud, two fog, and one edge layer.

RQ1: The `Local Runtime Monitor`[1], `Local Execution Adapter` *(together 96 LoC)*, and the initial *Architecture Configuration* (*47 LoC*) are use-case-specific; the remaining implementation is reusable for other use cases (*approx. 3h*). We created three use-case-specific SLs with increasing measures for a potential ICMP flood attack. The levels are implemented using the Python state machine package [21] (*70 LoC*, *approx. 1.5h*) and are defined as follows: Level 3 – Normal Readiness, Regular monitoring; Level 2 – Moderate Readiness, Rate limiting; Level 1 – Maximum Readiness, Block entirely. Each SL represents a state in the state machine. In the event of an attack, the `Local MAPE-K` triggers state transitions from one state to another. The stepwise transitions are in place for control and dependency management. SAFER-D fully supports configurable SL transitions (e.g., skipping a SL) when needed, which can be enabled by modifying the state machine definition. At runtime, each subsystem periodically issues heartbeat requests to check for the SLs of the other connected subsystems. For instance, `#2` sends a heartbeat every ten seconds (interval aligns with typical machine cycle times, allowing heartbeats to be sent alongside operational data), requesting the SLs of `#1`, `#6`, and `#4`. If one of the connected subsystems of interest responds with a higher criticality level than the one currently used for `#2`, `#2` adapts, i.e., transitions to the most critical SL. We prioritize the most restrictive SL to ensure rapid and effective response in high-impact scenarios like DoS attacks; in less time-critical contexts, incorporating human-in-the-loop decision-making can offer a more balanced trade-off between security and functionality. Furthermore, RESTful communication allows us to easily identify whether a device is unresponsive: If a request times out, the prototype adapts accordingly using global adaptations. The global adaptations are implemented by adding functionality to the periodic heartbeat checks. Each system sends out a tree traversal (REST calls), checking which subsystems are still reachable.

RQ2: We evaluated TtA_{SL} by monitoring our deployed framework operating on top of the edge computing architecture. Adaptations are checked every ten seconds via a heartbeat (i.e., `Local MAPE-K` and `Global MAPE-K` are periodically triggered). The goal is to capture the time from the beginning of an adaptation cycle (i.e., the start of the heartbeat) until an adaptation is detected (monitored), analyzed, processed, and executed. For instance, when considering Fig. 2a, `#1` is before the adaptation heartbeat in *Level 3*. `#2` is under attack and, therefore, in *Level 2*. For TtA_{SL}, we measure the time from the beginning of the heartbeat from `#1` until `#1` completes the adaptation to *Level 2*. – We chose `#1` as the monitored subsystem to control the influence of network depth: The closer the attacked subsystem is to the one monitored (in our case `#1`), the faster it can adapt. For instance, `#2` is closer to `#1` than `#8`. Since the heartbeat adaptation checks occur sequentially, network depth matters. Therefore, `#1` provides the best-case and worst-case scenario. It has systems connected directly (e.g., `#2`) and, at the same time, exhibits the longest depth to traverse. During the evaluation, we seeded predefined SL triggers for each system depicted in Fig. 2a. We measured the TtA_{SL} and validated that the adapted SLs were correct according to our seed, i.e., checking if `#1` transitions to *Level 2* when it is supposed to. We repeated this process for every subsystem $N = 100$ (i.e., a total of 700 SL adaptations). We employed a similar procedure for TtA_G. We randomly terminated components in Fig. 2a. The rest of the edge computing architecture had to respond accordingly and globally adapt to the PA mode. We measured the time again from the start of a heartbeat cycle until the system transitioned to the PA mode (i.e., all adaptation subgroups were formed). We chose `#1` again as our monitored component to control for network depth. Every subsystem in Fig. 2a is randomly terminated $N = 50$ times (350 in total).

4.3 Evaluation Results

RQ1: As a first step, we validated that every SL adaptation was carried out correctly as expected. Each change in the SL of an attacked subsystem in the architecture resulted in the expected change of `#1`. For the global adaptations, we can also confirm that SAFER-D's implementation executed correct adaptations in the architecture every time. Regarding the integration effort, we

[1]The specificity concerns the selection of monitored properties, not the underlying mechanism. Industrial systems often expose extensive metrics (e.g., Netdata), allowing the `Local Runtime Monitor` to function as a configurable filter or bridge

identified the components that would need re-implementation and counted the LoC and the time it took us to implement them. A total of 213 LoC are use-case-specific, which took us approximately 4-5h. Since every subsystem runs the same instance of SAFER-D, the development and configuration effort must be invested once. The code is pulled via version control on every subsystem and is ready to run without further configuration. Considering that the majority of SAFER-D's components can be reused for other use cases, 213 LoC and 4.5h of effort represent a considerably low effort with respect to the benefits SAFER-D can bring to such an architecture.

RQ2: Fig. 4a and Fig. 4b show the boxplots of our quantitative analysis. One can notice the considerably long adaptation times for TtA_G, especially compared to TtA_{SL}. However, 3 seconds of the values in TtA_G can be attributed to the initial HTTP timeout. An HTTP request always waits 3 seconds (the time an SL adaptation would take additionally) for a response during the heartbeat. After the waiting period, the system is deemed unresponsive. Therefore, these numbers always contain a fixed constant of 3000 ms. The adaptation times for the TtA_{SL} remain consistent across runs (cf. distribution in Fig. 4a). The average median of TtA_{SL} is 344.86 ms (roughly comparable to the 333 ms in [25]). Therefore, most of TtA_{SL} are considerably lower than half a second. Similarly, for TtA_G, the average median adaptation time is 4543.29 ms, which means 1543.29 ms (total minus 3 seconds timeout) solely for the adaptation. However, adaptation times are still longer compared to TtA_{SL}. The reason is that once a subsystem is unresponsive, the system again traverses through the tree to determine the subgroup. The traversal takes time, as shown in TtA_G. The distribution of adaptation times among the subsystems can be considered equally stable for all subsystems for TtA_{SL}. For TtA_G, the distribution is dense for `#6`, `#4`, `#5`, and `#8`. `#2` and `#3` yielded a rather loose distribution, although their median is similar to the other subsystems.

4.4 Answers to Research Questions

Answer to RQ1: The use case was inspired by a real-world industrial example. SAFER-D was successfully implemented within the edge computing architecture similar to the one utilized by the partnering company. This application validated the framework's capability to address practical challenges in an industry-relevant context. Although not discussed here, due to space constraints, we deployed SAFER-D to a second use case focused on web authentication, further showcasing SAFER-D's adaptability to different domains and highlighting its ability to cater to security requirements in diverse settings (details can be found in the supplemental material; cf. data availability). Both prototypical implementations are publicly available, enabling users to explore and leverage the adaptation mechanisms provided by SAFER-D. The implementation effort for SAFER-D's adaptation components in our scenario offers a first indication of manageable integration (considering time and LoC), though further validation is needed. A total of 21h were invested to implement `Global MAPE-K` components and the `Local Adaptation Middleware` and `Local SL Manager` (i.e., the reusable parts of SAFER-D); the `Local Runtime Monitor` and `Local Execution Adapter` are use-case-specific and took us approx. 4.5h. We found the PA mode particularly helpful during development since even a network misconfiguration immediately resulted in respective global adaptations. For this reason, we find SAFER-D's global adaptations helpful not only in the event of a security incident but also in maintenance or operational malfunctions. In summary, through these applications, we demonstrated SAFER-D's applicability in real-world scenarios. The initial results indicate that integration is feasible; however, a more comprehensive evaluation is required for general claims.

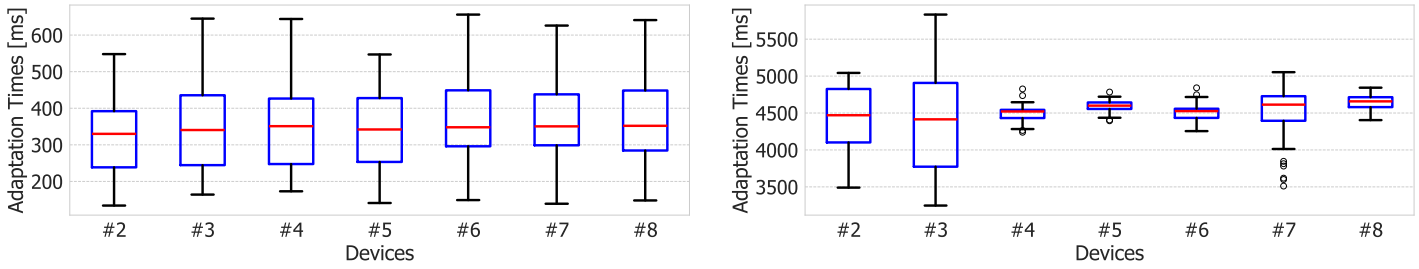
Answer to RQ2: We comprehensively evaluated SAFER-D's efficiency by quantitatively analyzing data from our use case, measuring adaptation times for SL adaptations (TtA_{SL}) and global adaptations (TtA_G). Adaptation times were sufficiently efficient for the given scenarios. Our analysis suggests that the centrality of the subsystem being considered influences adaptation times across the architecture (cf. Fig. 4b). In general, top-level systems require longer adaptation times compared to low-level systems. Optimized implementation strategies can improve performance, such as parallel tree traversal for adaptation checks. While our findings highlight areas with potential for further optimization, the prototypical implementations provide a proof of concept. The results affirm that SAFER-D delivers promising efficiency, making it a viable framework for dynamic system adaptations in distributed computing systems.

5 Threats to Validity

Like any study, our work faces threats to validity. For conclusion validity, we used quantitative metrics TtA_{SL} and TtA_G and repeated runs to ensure consistency, though relying solely on time-based measures limits insight into security–functionality trade-offs; broader quantitative applicability metrics could offer a more comprehensive picture. Regarding internal validity, we controlled the experimental setup to isolate the framework's effect on adaptation times, minimizing the influence of hardware / network factors, although real-world deployments may introduce unforeseen variables. For external validity, our use cases and metrics (e.g., time, LoC) serve as a foundation for generalization; however, further validation in diverse environments is necessary to confirm broader applicability.

6 Related Work

Multi-level Adaptation: In their work, Jahan *et al.* [13] propose a framework for dynamically maintaining functional and security concerns in autonomous systems, ensuring coordination between multiple MAPE-K feedback control loops. An additional MAPE-SAC loop is introduced that emphasizes security-related adaptations. Similarly, also employing a multi-feedback loop approach, Vromant *et al.* [31] relied on intra-loop and inter-loop coordination of multiple MAPE-K loops to perform coordinated adaptation actions. Braberman *et al.* [3] present MORPH, a reference architecture for self-adaptation based on the MAPE-K loop. MORPH consists of three layers for goal management, strategy management, and strategy enactment with different reconfiguration strategies. Ben Halima *et al.* [2] introduce a set of MAPE-K



(a) Boxplot for TtA_{SL}.

(b) Boxplot for TtA_{G}.

Figure 5: Boxplots of TtA_{SL} & TtA_{G}.

design patterns tailored for decentralized control in self-adaptive CPSs. Gerostathopoulos *et al.* [11] propose IRM-SA, an Invariant Refinement Method for Self-Adaptation, tailored to ensure dependability and adaptivity in software-intensive CPSs.

Security Adaptation: Fotohi *et al.* [10] propose an Agent-based Self-Protective method (ASP-UAVN) inspired by the human immune system to enhance secure communication in Unmanned Aerial Vehicle Networks. Riegler *et al.* [25] introduce DSEC4IoT, a distributed MAPE-K framework for self-protective IoT devices, enabling local and centralized monitoring, analysis, planning, and execution of security measures. Jones *et al.* [14] present Crispy, a CRISPR-inspired (bacterial adaptive immune system) resiliency mechanism to protect N-variant systems from DoS attacks by leveraging automatic attack signature generation. Finally, Skandylas [27] presents an approach for enhancing adaptive security in software-intensive systems by equipping them with self-protective capabilities, including runtime threat modeling, proactive adaptation, and decentralized trust-based mechanisms.

We provide an overview of the addressed contents of related work and how it compares to SAFER-D in Table 1. The table reveals that SAFER-D is closely related to Fotohi *et al.* [10] and Riegler *et al.* [25]. – Fotohi *et al.* [10] focus on *securing communication* between UAVs, relying on detecting network-layer attacks and isolating malicious nodes. In contrast, SAFER-D *protects the devices themselves* through hierarchical coordination (vs. purely peer-to-peer) and adaptive security levels (vs. strict isolation), enabling graded and context-aware responses. Unlike Riegler *et al.* [25], who rely on a central "Managing Server" to coordinate security adaptation across independently operating devices, SAFER-D enables fully decentralized coordination among autonomous subsystems. Moreover, DSec4IoT supports a fixed two-level structure (server and devices) while SAFER-D introduces a multi-level hierarchy where adaptation decisions can propagate and adjust across layers. Therefore, in case of connection loss, SAFER-D can adapt within subgroups while DSec4IoT adapts only locally.

7 Conclusion

In this paper, we presented SAFER-D, a novel self-adaptive security framework for DCAs. SAFER-D integrates diverse adaptation strategies to enable security adaptations, even in the event of system failures caused by attacks. Our evaluation using a realistic use case scenario has confirmed that SAFER-D can be used in practice and that the adaptations are efficiently carried out. As part of our ongoing and future work, we aim to extend our adaptation strategies, improve performance and scalability for large-scale architectures, and incorporate advanced runtime threat modeling techniques.

Data Availability

Supplemental material on GitHub: Edge computing use case / WebAuthn use case.

References

[1] D. Acarali, K. Rajesh Rao, M. Rajarajan, D. Chema, and M. Ginzburg. Modelling smart grid IT-OT dependencies for DDoS impact propagation. *COSE*, 112:102528, Jan. 2022.
[2] R. Ben Halima, M. Hachicha, A. Jemal, and A. Hadj Kacem. MAPE-K patterns for self-adaptation in cyber-physical systems. *The J. of Supercomputing*, 79(5):4917–4943, Mar. 2023.
[3] V. Braberman, N. D'Ippolito, J. Kramer, D. Sykes, and S. Uchitel. MORPH: A reference architecture for configuration and behaviour self-adaptation. In *Proc. of the 1st Intl. Workshop on Control Theory for Software Eng.*, pages 9–16, Aug. 2015.
[4] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, 2010.
[5] K. Cao, Y. Liu, G. Meng, and Q. Sun. An Overview on Edge Computing Research. *IEEE Access*, 8:85714–85728, 2020.
[6] A. Carlo and K. Obergfaell. Cyber attacks on critical infrastructures and satellite communications. *Intl. J. of Critical Infrastructure Protection*, 46:100701, Sept. 2024.
[7] T. Chambers, J. Cleland-Huang, and M. Vierhauser. Self-Adaptation of Loosely Coupled Systems across a System of Small Uncrewed Aerial Systems. In *Intl. Workshop on Software Eng. for SoS and Software Ecosystems*, pages 37–44, Aug. 2024.
[8] L. Coppolino, S. D'Antonio, R. Nardone, and L. Romano. A self-adaptation-based approach to resilience improvement of complex internets of utility systems. *Environment Systems and Decisions*, 43(4):708–720, Dec. 2023.
[9] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C.-T. Lin. Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment. *IEEE Access*, 6:1706–1717, 2018.
[10] R. Fotohi, E. Nazemi, and F. Shams Aliee. An agent-based self-protective method to secure communication between UAVs in unmanned aerial vehicle networks. *Vehicular Communications*, 26:100267, Dec. 2020.
[11] I. Gerostathopoulos, T. Bures, P. Hnetynka, J. Keznikl, M. Kit, F. Plasil, and N. Plouzeau. Self-adaptation in software-intensive cyber–physical systems: From system goals to architecture configurations. *JSS*, 122:378–397, Dec. 2016.
[12] D. He, H. Gu, S. Zhu, S. Chan, and M. Guizani. A Comprehensive Detection Method for the Lateral Movement Stage of APT Attacks. *IEEE Internet of Things J.*, 11(5):8440–8447, Mar. 2024.
[13] S. Jahan, I. Riley, C. Walter, R. F. Gamble, M. Pasco, P. K. McKinley, and B. H. C. Cheng. MAPE-K/MAPE-SAC: An interaction framework for adaptive systems with security assurance cases. *Future Generation Comp. Sys.*, 109:197–209, 2020.
[14] J. Jones, J. D. Hiser, J. W. Davidson, and S. Forrest. Defeating denial-of-service attacks in a self-managing N-variant system. In *Proc. of the 14th Intl. SEAMS*, pages 126–138, May 2019.
[15] S. Kaplan and B. J. Garrick. On The Quantitative Definition of Risk. *Risk Analysis*, 1(1):11–27, 1981.
[16] J. Kephart and D. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, Jan. 2003.
[17] R. D. Larkin, J. Lopez, J. W. Butts, and M. R. Grimaila. Evaluation of security solutions in the SCADA environment. *SIGMIS Database*, 45(1):38–53, Mar. 2014.
[18] R. Leszczyna. Aiming at methods' wider adoption: Applicability determinants and metrics. *Computer Science Review*, 40:100387, May 2021.

	[13]	[31]	[3]	[2]	[11]	[10]	[25]	[14]	[27]	SAFER-D
Multi-System	◑	◑	○	●	●	◑	●		●	●
Security Focus	●			○		●	●	●	●	●
Failure Resilience		●	◑	○	●	◑	●	●		●
Loop Interaction	●	●	●	●	●	◑	◑	○		●
Decentral Coordination		◑		●	●	◑	○		●	●

Table 1: Overview of Related Work; [ ]=Aspect absent, [○]=Aspect mentioned, [◑]=Aspect partially addressed, [●]=Aspect fully addressed

[19] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung. Intrusion detection system: A comprehensive review. *J. of Netw. and Computer App.*, 36(1):16–24, Jan. 2013.
[20] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi. Edge Computing for Autonomous Driving: Opportunities and Challenges. *Proc. of the IEEE*, 107(8):1697–1716, Aug. 2019.
[21] F. Macedo. fgmacedo/python-statemachine, Dec. 2024.
[22] J. Momoh. *Smart Grid: Fundamentals of Design and Analysis*. Wiley, Mar. 2012.
[23] Y. Otoum and A. Nayak. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. *J. of Network and Systems Management*, 29(3):23, July 2021.
[24] V. R. Palleti, S. Adepu, V. K. Mishra, and A. Mathur. Cascading effects of cyber-attacks on interconnected critical infrastructure. *Cybersecurity*, 4(1):8, Mar. 2021.
[25] M. Riegler, J. Sametinger, and M. Vierhauser. A Distributed MAPE-K Framework for Self-Protective IoT Devices. In *Proc. of the 18th Intl. SEAMS*, pages 202–208. IEEE, May 2023.
[26] P. K. Sadhu, V. P. Yanambaka, and A. Abdelgawad. Internet of Things: Security and Solutions Survey. *Sensors*, 22(19):7433, Sept. 2022.
[27] C. Skandylas. Design and analysis of self-protection : Adaptive security for software-intensive systems. In *Companion Proc. of the 15th European Conf. on Software Architecture: ECSA 2021 Companion*. CEUR-WS, 2021.
[28] M. Stadler, M. Riegler, and J. Sametinger. Cyber-Resilient Edge Computing: A Holistic Approach with Multi-Level MAPE-K Loops. In *2024 IEEE 21st Intl. Conf. on Software Architecture Companion (ICSA-C)*, pages 79–83. IEEE, June 2024.
[29] K. Thakur, M. L. Ali, N. Jiang, and M. Qiu. Impact of cyber-attacks on critical infrastructure. In *2016 IEEE BigDataSecurity, HPSC, and IDS*, pages 183–186, 2016.
[30] T. Theisen. DEFCON Levels, Jan. 2023.
[31] P. Vromant, D. Weyns, S. Malek, and J. Andersson. On interacting control loops in self-adaptive systems. In *Proc. of the 6th Intl. SEAMS*, pages 202–207, 2011.
[32] D. Weyns and J. Andersson. On the challenges of self-adaptation in systems of systems. In *Proc. of the First Intl. Workshop on Software Eng. for Systems-of-Systems*, pages 47–51, July 2013.
[33] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka. On Patterns for Decentralized Control in Self-Adaptive Systems. In *Software Engineering for Self-Adaptive Systems II*, pages 76–107. 2013.
[34] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv. Edge Computing Security: State of the Art and Challenges. *Proc. of the IEEE*, 107(8):1608–1631, Aug. 2019.
[35] O. Yoachimik and J. Pacheco. Record-breaking 5.6 Tbps DDoS attack and global DDoS trends for 2024 Q4, Jan. 2025.