– Preprint –
Accepted for publication at the SEAMS 2023. Final published version available at:
https://doi.org/10.1109/SEAMS59076.2023.00034

A Distributed MAPE-K Framework for Self-Protective IoT Devices

Michael Riegler*†, Johannes Sametinger*†, Michael Vierhauser*
*LIT Secure and Correct Systems Lab
†Department of Business Informatics - Software Engineering
Johannes Kepler University Linz
Linz, Austria
firstname.lastname@jku.at

***Abstract*—Internet of Things (IoT) devices have become ubiquitous in our everyday life, with security becoming an ever-growing issue as more and more cyber-attack incidents being reported, primarily due to deficiencies in existing security mechanisms. However, while, for example, cloud-based applications, or industrial automation systems of systems possess significant resources for monitoring health, and determining their status and correct behavior at runtime, IoT devices operate with limited hardware capabilities and under tight resource constraints, making monitoring, analysis, and response activities a challenging endeavor. Following the *NIST Cybersecurity Framework*, IoT devices need to *identify*, *protect*, *detect*, *respond*, and *recover* from cyber-attacks, unauthorized access, and other security threats. A common way to provide self-adaptation to changing conditions is the MAPE-K loop with four pivotal phases: Monitor, Analyze, Plan, and Execute. This paper presents DSec4IoT, a "Distributed MAPE-K Framework for Self-Protective IoT Devices". Our framework leverages the idea of distributed MAPE-K patterns and establishes a model for managing and controlling *Self-Protective IoT Devices*. We evaluate our approach by simulating port scans and performing adaptation activities. Results have confirmed that DSec4IoT can be easily applied to detect and mitigate them.**

***Index Terms*—IoT, Security, MAPE-K Loop, Self-Protecting, Mode Switching**

I. Introduction

The number of worldwide connected *Internet of Things* (IoT) devices is projected to exceed 29 billion by 2030 [1] – more than double compared to 2022. Example applications range from consumer internet and media devices [2], smart grids [3], [4], inventory management, to connected vehicles [5]. The market value of industrial IoT platforms for manufacturing will increase to 22.3 billion USD by 2025 [6].

In this context, security has been an ever-growing issue, with more and more reported cyber-attacks, which can be primarily attributed to inadequate security mechanisms. Security is one of the biggest challenges for IoT networks [7], [8], with IoT devices being used for *Distributed Denial-of-Service* (DDoS) attacks as part of botnets, and dealing with network intrusion and unauthorized access.

While cloud-based applications or industrial automation systems possess significant resources for monitoring health and determining their status and correct behavior at runtime, IoT devices operate with limited hardware capabilities and tight resource constraints, making monitoring, analysis, and response activities a challenging endeavor [9]. Depending on the type of security issue, or system state, an analysis may need to be performed locally on the device, or centrally by a higher-level authority [10]. Following the *NIST Cybersecurity Framework* [11], IoT devices need to *identify*, *protect*, *detect*, *respond*, and *recover* from cyber-attacks, unauthorized access, and other threats. We need monitoring capabilities, but more importantly, sophisticated mechanisms for analyzing information from different sources, and subsequent (automated) execution of appropriate actions to mitigate security threats.

To address these issues, and to deal with security threats at various levels of granularity, we leverage self-adaptation mechanisms [12] with the *MAPE-K* feedback loop [13]. Industrial applications use *MAPE-K* for optimizing performance, automating tasks, detecting potential threats, and protecting systems [14]. Typically, these systems automatically perform tasks to achieve a specific goal. In our work, we focus on *Security* and the self-* aspect of protection [15], [16]. We propose a "Distributed MAPE-K Framework for Self-Protective IoT Devices" (DSec4IoT), building on the idea of the distributed *MAPE-K* loop [17], leveraging these patterns and establishing a model for managing and controlling in the context of *Self-Protective IoT Devices*.

Our framework implements the four phases of the *MAPE-K* loop: Monitor, Analyze, Plan, and Execute at a managing (server) and IoT-device level. It consists of an *Intrusion Detection and Prevention System* (IDPS) on the IoT devices, and a *Complex Event Processing* (CEP) engine on the server. The framework provides capabilities to deploy and configure the *IDPS* and the firewall, filters to observe events, constraints to analyze events, and actions to adapt the behavior accordingly. Self-protection in this context means that both the server and the individual IoT devices can perform activities partly or fully automated without human intervention.

We claim the following contributions:

- A self-protective solution for IoT devices that monitors and mitigates online and offline attacks by sharing attacker information and context-aware adaption of its behavior.
- The instantiation of a distributed *MAPE-K* loop in the case of IoT environments and an empirical evaluation of the applicability and performance of the adaptive solution with one common attack type.

– Preprint –
Accepted for publication at the SEAMS 2023. Final published version available at:
https://doi.org/10.1109/SEAMS59076.2023.00034

In this paper, we present challenges and a motivating example in Section II. We then introduce our distributed framework in Section III and evaluate our approach by applying it to a use case involving various IoT devices in Section IV. Finally, we discuss results, threats to validity, and related work in Sections V – VII, and conclude with future work in Section VIII.

II. Challenges and Motivating Example

Securing IoT devices is challenging due to resource constraints, such as limited storage, memory, CPU, and power [18], [19]. However, IoT devices are exposed to various attacks, such as bots and automated attack scripts trying to break into these devices and guess credentials with dictionary and brute-force attacks [19]. Common countermeasures in such cases are, for example, delaying login attempts, or blocking accounts and IP addresses after several wrong attempts [20], [21].

Fig. 1 depicts a simple example. On the left side, the IoT devices are managed independently (locally on the device only), and do not share any information with a central server. Implementing a basic security concept, attackers are blocked on that device after two failed login attempts. However, with no central managing authority, each IoT device stands alone, and attackers can perform the same number of attempts on each device (1a-3a).

On the right side of Fig. 1, IoT devices share the attackers' information with a higher-level authority, i.e., a server. The server can consolidate information from multiple devices, and thus recognize attacks from one attacker on multiple IoT devices (1b-2b). Once an attack has been detected, the server can inform other connected devices to automatically block the attacker and prevent further attacks (3b). In the scenario with three IoT devices ($n = 3$), sharing attackers' information halves the possible brute-force attempts from six ($2 * n$) to a maximum of three. The results are even more severe in the real world, where dozens or even hundreds of devices are typically deployed. For example, with 100 devices ($n = 100$), an attacker would have 200 ($2 * n$) attempts on stand-alone devices. Sharing information limits attackers to only three attempts, which means a reduction of 98.5% for $n = 100$.

The scenario is exacerbated when the attack is distributed, and carried out by multiple attackers (DDoS) [22]. Blocking them individually will no longer suffice, even if IoT devices share attacker information. Each IP address can make three attempts before it is blocked. So if we have 100 devices ($n = 100$) and 100 coordinated attackers ($a = 100$), the attackers have 300 ($3 * a$) attempts. If they focus on one IoT device, the chances of brute-forcing a password are even higher. Therefore, devices must be armed with multiple modes that can be activated and switched on demand [23], [24]. Each mode hereby provides specific functionality and has a specific attack surface. The server, as well as the IoT device itself, can trigger a mode switch. For example, in the *Blocklist-Mode*, the device accepts connection attempts from any IP address, but after two wrong attempts, the IP address is added

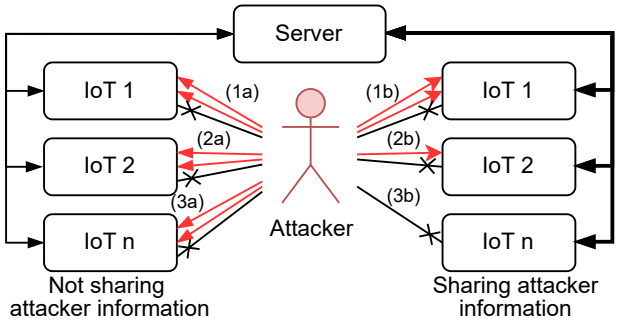

Fig. 1: IoT device example with information being shared among devices, vs. not sharing information.

to the blocklist and banned from further attempts. Sharing the attacker's information with a server and other IoT devices can enrich this mode. However, the blocklist may overflow if there are hundreds or even thousands of blocked IP addresses. Consequently, switching to the *Allowlist-Mode*, which allows only predefined IP addresses to connect, changes the behavior and the attack surface. This mode provides less flexibility but more security.

Another aspect is represented by vulnerabilities [23]. Once detected, attackers can exploit them until a patch is provided and installed. Analyzing the vulnerabilities' status on individual IoT devices is impractical. On the one hand, we need to detect local incidents and share them with a central server. On the other hand, we can reduce the overall attack surface by centrally collecting and analyzing information about local incidents, vulnerabilities, and patches and by sharing this information with connected devices. This strengthens the need for distributed security. But for IoT devices, simply monitoring and sending events to a server is insufficient. If an IoT device has no connection, e.g., due to a deliberate attack, the server cannot relay information and trigger a mode switch or adaptation, yet the device should still be able to protect itself.

III. The DSec4IoT Framework

Based on the challenges identified in Section II, we have derived several requirements for adapting to security threats and dynamic mode switching for IoT devices.

First, IoT devices require lightweight data exchange protocols to forward information about potential attacks and attackers. The server can analyze this information from multiple devices, plan countermeasures, and send execution actions to all connected devices. Then operators or IT security professionals can make decisions on the server side (human-in-the-loop). Furthermore, IoT devices must be able to detect and analyze events and suspicious behavior independently, even offline or without connection to the server. For this purpose, we employ *Event-Condition-Action* (ECA) rules, where an event triggers a rule, then the condition is checked, and if met, predefined actions are executed. When the device regains connection, it can forward any event that occurred in the meantime.

– Preprint –
Accepted for publication at the SEAMS 2023. Final published version available at:
https://doi.org/10.1109/SEAMS59076.2023.00034

A. Framework Overview

DSEC4IOT uses decentralized control patterns proposed by Weyns et al. [17] at the server and IoT devices level, with different levels of abstraction. We focus on self-protecting activities and adopt a combination of the *Hierarchical Control Pattern* (HCP) and IBM's architectural blueprint [25], which separates the complexity of self-adaption into several layers. Each IoT device hereby has its local environment, such as a company or manufacturing network, while the server has a more global/strategic perspective. To coordinate this, the M and E components of the server interact with these components at the IoT device level. Fig. 2 provides an overview of our *Self-protective IoT System Architecture*.

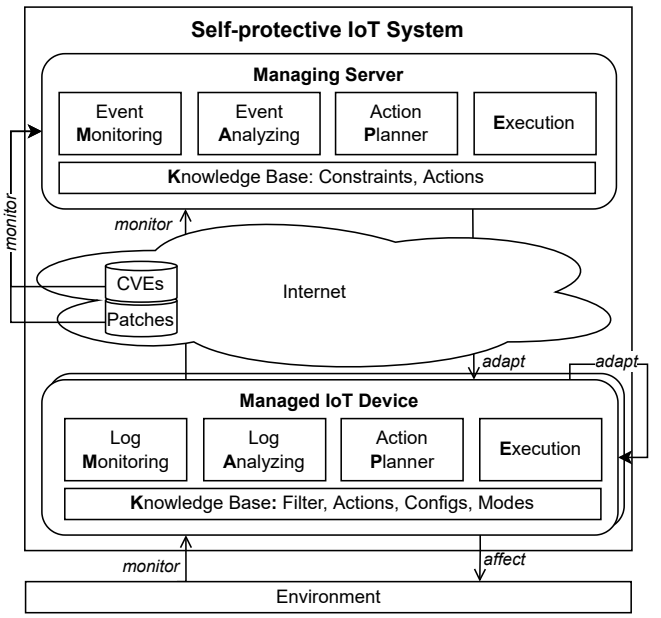

Fig. 2: DSEC4IOT Framework: A Self-protective IoT System Architecture

• **Managing Server:** This building block implements the four activities of the *MAPE-K* control loop and stores constraints and actions in an ECA rules-based *Knowledge Base*. The *Event Monitoring* component collects events from the *Managed IoT Devices* and forwards them to the *Event Analyzer*, which in turn checks them against specified rules and provides information to the *Action Planner* if a condition is fulfilled. As part of these global checks, it is necessary to analyze not only individual events and the data attached to them but, more importantly, streams of events and temporal constraints, which are used to recognize patterns and detect security issues across various resources or time. The *Action Planner* considers the context, like the offline state, and selects appropriate actions from the *Knowledge Base* to adapt the *Managed IoT Devices* to meet security goals and notifies the *Executor* to execute those actions on all, on some, or on only affected *Managed IoT Devices*. Examples of actions include sending notifications to the administrator, executing shell scripts, sending emails, sending HTTP requests, blocking IP addresses, or starting/stopping services. The *Managing Server* can also establish two-way communication with the *Managed IoT Devices* to request their current state. Besides that, the *Managing Server* leverages additional sources of information such as *Common Vulnerabilities and Exposures* (CVEs), and updates/patches of the used operating systems, applications, and libraries from public databases like the *National Vulnerability Database* [26].

• **Managed IoT Device:** Each IoT device also implements the full range of *MAPE-K* control loop activities to prevent delays and protect offline devices. Attacks, such as DDoS, may lead to performance issues and disrupt the connection to the server. DSEC4IOT focuses on IoT device level *Log Monitoring* with an emphasis on security and self-protection. Dedicated *Filters*, with specific characters or patterns from the *Knowledge Base*, continuously observe the system logs, network interfaces, and critical applications for suspicious entries. These filters can log events at the hardware level, such as newly attached USB devices, connected network cables, or frequent restarts. At the software/system level, information related to authentication errors (i.e., failed authentication attempts that may occur locally or via the network), port scans, and HTTP errors can be monitored and collected. We provide further examples of filters and their application in Section IV. When a filter is applied, the information is transformed into an event and a dedicated message notifying the *Log Analyzer*. This component reviews various sources of IoT-device information to monitor the frequency and source of suspicious log entries and inform the *Action Planner* if limits w.r.t. maximal login attempts are exceeded. The *Action Planner* selects predefined countermeasures from the *Knowledge Base* and sends them to the *Executor*. Countermeasures include blocking an IP address, adapting configurations, switching the operating mode, or rebooting the system, which may affect the device and its environment.

• **Transfer of Knowledge & Human-in-the-loop:** One important aspect when managing multiple distributed IoT devices is the administration of local rules, conditions, and actions. While this information has to be available locally (to enable the device to perform tasks independently), we administrate the knowledge bases for both the *Managing Server* and the *Managed IoT Devices* on the *Managing Server*. Filters, constraints, actions, configurations, modes, and other settings can be created and maintained centrally. After tests with single devices, we deploy them to the individual *MAPE-K* loops of the IoT devices with infrastructure automation tools. Another aspect is the integration of the human, e.g., a server administrator or engineer. While preventing potential attacks by blocking an attacker's IP address has to be performed automatically to avoid delays and efficiently react to attacks, other actions may require human confirmation or input. Before stopping a remotely *Managed IoT Device* and annoying customers, the operator (human-in-the-loop) should confirm this action. In the case of frequent events and precise knowledge of the necessary actions, this process can be partly- or fully automated.

– Preprint –
Accepted for publication at the SEAMS 2023. Final published version available at:
https://doi.org/10.1109/SEAMS59076.2023.00034

IV. Empirical Evaluation

To demonstrate the applicability of our approach and assess the potential benefits, we have conducted a case study where we applied our DSec4IoT framework for monitoring, analyzing, and updating components in response to different security threats. We followed the guidelines described by Runeson and Hoest [27] for defining use case requirements and specifying the evaluation setup and metrics. We explore the following *Research Questions* (RQs):

RQ1 (Applicability): Can DSec4IoT *be used for defining and executing realistic test scenarios, and what is the effort required for creating and specifying them?*

We investigate to what extent our framework can be used in a realistic case study setting and assess the effort required for creating and deploying monitoring components and specifying server and IoT-side rules and adaptations.

RQ2 (Performance): Is DSec4IoT *capable of detecting different types of security issues, and how long does it take to apply a countermeasure?*

We assess the suitability for handling different types of security issues, evaluating the performance to detect, analyze, and respond to attacks.

Case Requirements: To assess DSec4IoT's general applicability and its performance, a suitable test scenario with an appropriate infrastructure is necessary. This involves three main components: (1) a *Managing Server*, (2) multiple *Managed IoT Devices* connected to a network that can be subject to potential attacks and security threats, and (3) simulation of attacks as described in previous research [28], [29]. Both local and cloud analysis and execution of adaptations are required, as well as logging and analyzing events to evaluate the performed execution actions.

Selected Use Case: Based on these requirements, we created ECA rules (cf. Table I) and discussed several test scenarios for our DSec4IoT framework at the IoT device and managing server level. For each event, we specified the number of wrong attempts and a monitoring window in the condition. Additionally, we created actions to mitigate and prevent further attacks, e.g., blocking the attacker's IP, increasing the waiting time, rebooting the system, and adapting security policies.

We selected *port scans* for our use case because attackers typically use tools like *Nmap* [30] during the reconnaissance phase [31] to scan for open ports on the network layer and running services. Then, they search for any potential service vulnerability to attack them. Timely and fast detection and execution of appropriate countermeasures and necessary system adaptations are of key importance. We utilize the *MAPE-K* loop to monitor events related to security threats, analyze them through specific conditions, and plan and execute actions to mitigate and prevent further attacks.

Evaluation Settings: For our evaluation, we created a testbed that simulates a network with multiple connected IoT devices. For this purpose, we set up the test environment containing five *Raspberry Pi* (RPi) *4 Model B* (1.5 GHz quad-core ARM Cortex-A72, 4 GB RAM, Ubuntu 22.04.1 LTS) and three *RPi Zero 2 W* (1 GHz quad-core, 64-bit ARM Cortex-A53, 512 MB RAM, Raspbian 11).

TABLE I: IoT Device and Managing Server Test Scenarios for our MAPE-K Security Framework

ID	Event	Condition	Action
IoT Device Scenarios			
D01	*Port scan*	$\geq$ 2 attempts in 1 min	block IP for 10 min
D02	*SSH login failure*	$\geq$ 2 attempts in 5 min	block IP for 5 min
D03		$\geq$ 4 attempts in 1 hour	block IP for 1 hour
D05	*TTY login failure*	$\geq$ 2 attempts in 5 min	waiting time
D06		$\geq$ 4 attempts in 1 hour	reboot system
D08	*HTTP errors*	$>$ 2 attempts in 5 min	block IP for 5 min
D12	*Get/Lost con.*	every attempt	adapt security
Managing Server Scenarios			
S01	*Blocked IP*	$>$ 2 attempts in 5 min from $\geq$ 2 devices	block IP on all devices for 1 hour
S02	*IoT changed public IP*	every attempt	inform admin

An RPi 4 was used as *Managing Server* running our *Java Server Application* with the *OpenJDK Runtime Environment* (build 18.0.2), including the *MQTT*-Broker (Eclipse Mosquitto v2.0.11) [32] for monitoring events of the IoT devices and the *Esper CEP Engine* (4.7.0) [33] to analyze events and check constraints using the *Event Processing Language* (EPL). *Ansible* (2.10.8) [34] was used for automation, orchestration and executing actions. Another RPi 4 was used as *Attacker*. The remaining six RPis represent the *Managed IoT Devices* and had an *OpenSSH-Server* (8.4-8.9), *Fail2ban* (F2b) (0.11.2) [35] as local IDPS, and the *Mosquitto Client Package* (2.0.11) for sending events installed and running.

Evaluation Metrics and Data Collection: For RQ1, we measured the human effort to use the framework and analyzed the applicability. We created a shell script to measure the deployment and installation and executed it ten times to calculate average values. For RQ2, we used common security and penetration test tools on the *Attacker Device*. We built a shell script to randomly select IoT devices, simulate attacks, measure timestamps, and document results in a log file. We subsequently analyzed the log files on the *Attacker Device*, the *IoT Devices*, and the *Managing Server*, and measured the duration for detection and response. To obtain more precise results in milliseconds (ms), we altered the *Rsyslog* configuration. Before each run, we updated the time on all devices to ensure accurate measurements.

A. RQ1: Applicability

We took the following measures to identify and respond to port scans. To prevent additional attacks on the IoT device during the reconnaissance phase, we propose discouraging potential attackers who perform port scans by temporarily

– Preprint –
Accepted for publication at the SEAMS 2023. Final published version available at:
https://doi.org/10.1109/SEAMS59076.2023.00034

blocking their IP address for a specific duration (e.g., 10 minutes). Typically, requests during a port scan to unused ports are answered by the system with closed or denied and are not logged. To monitor these scans, we extended the IoT devices firewall (iptables) with additional IP packet filter rules. We created a user-defined chain PORTSCAN, appended it to the bottom of the INPUT chain, and filtered and logged every request to unused ports before we dropped it. Possible attackers get no reply, but this mechanism allows us to monitor the system logs with a regular expression filter, analyze port scans, and identify the attacker's IP address.

To plan and execute corresponding actions for blocking and relaying the information to the server, we defined a *F2b* jail (see LISTING 1). The configuration includes the filter (portscan), the log path, and the actions to run if the conditions are met. These are the log monitoring window (findtime) and the number of found entries (maxretry).

Listing 1: IoT Device: Excerpt of an Fail2ban ECA Rule for detection of and respond to Port Scans

```
- name: portscan
  logpath: /var/log/kern.log
  action: |
    iptables-allports[blocktype=DROP]
    publish_mqtt_message[topic="portscan",message
        ="{\"ip\":\"<ip>\"}"]
    offline-adaptation
  maxretry: 2
  findtime: 1m
  bantime: 10m
```

If there is a match, *F2b* runs the pre-defined actions: "iptables-multiport" to ban the IP temporarily for 10 minutes (bantime), "publish_mqtt_message" to notify the server per MQTT message, and "offline-adaptation" to check the connection state to the server and make adaptions based on that.

Sharing Attacker Information: In order to support shared attacker information from the *Managing Server*, we extended the firewall of the IoT devices by creating a user-defined chain BLOCKLIST. IP addresses on this list are blocked, and any further access attempts are rejected.

IoT Device Offline Adaptation: If the server connection is lost and attacks are ongoing, neither the automatic monitoring and control solutions on the server nor an administrator can intervene. Therefore, we have equipped the IoT devices with self-protection methods, switching the devices to a more restrictive mode and increasing the ban time exponentially for each wrong attempt. If the number of banned IP addresses exceeds a predetermined value, the system switches to *Allowlist-Mode*, and only trusted IP addresses are granted access. Once the banned IP addresses fall below a certain value, the system switches back to *Blocklist-Mode*. The administrator is notified in both cases and can take further action.

Managing Server: Before we deployed the previously described configurations and adaptions for the attack scenario to all IoT devices, we manually tested the applicability on a single device. The testing involved monitoring the "portscan" topics with an MQTT Subscriber and creating a "BlockedIpEvent" based on incoming messages. EPL constraints (see LISTING 2) were specified to analyze these events and select blocked IP addresses that were blocked on more than two different devices within five minutes. If the constraint is violated, an Ansible playbook is executed to add the IP address to the user-defined chain BLOCKLIST on the IoT device's firewall.

Listing 2: Server: EPL Constraint used with Esper to detect IP adresses blocked on multiple hosts

```
NAME = SameBlockedIpFromMoreThanOneHost
CONSTRAINT =
   select * from BlockedIpEvent.win:time(5 min)
       group by ip having count(*) >= 2
```

Results: With our custom *Ansible F2b* module, we could reduce the basic configuration to 15 lines of code (LoC). The initial automated deployment of the software, the configuration files, and scripts to the seven IoT devices took less than 5 min (approx. 30 sec per device), including updating time, installing *mosqitto*, configuring the firewall, and installing/configuring *F2b*. Checking the deployment on all devices was much faster and took only around 3 minutes. The human effort required for implementation, testing, and creating filters, actions, configurations, modes, and deployment to the IoT devices was approximately 30 hours for one author.

Regarding RQ1 and the applicability of our DSEC4IOT framework, we can conclude that we were able to create all necessary artifacts for the portscan scenario and could reduce the overall effort compared to configuring all nine IoT devices manually. The deployment was reduced to less than 5 minutes.

B. RQ2: Performance

In RQ2, we assessed how well DSEC4IOT responds to attacks by measuring the performance. We ran 100 port scans on the IoT devices with our attacker script using *Nmap 7.80* with a 60-second pause between each run. We examined the logs on the *Attacker Device*, the kernel logs and the *F2b* log entries on the IoT devices, and the logs on the *Managing Server*. Fig. 3 shows the average detection and reaction time.

Based on this comparison, we can conclude that our *F2b* filter pattern detected all port scans. However, the monitoring process exhibited a slight delay, with the *F2b* log timestamps being 179ms behind the kernel log entries.

Initially, it took the attacker script 200ms per IoT device to scan ten ports before getting banned. The IoT devices took, on average, 86ms to monitor and analyze the port scan and 247ms to plan and execute predefined actions (333ms in total). The

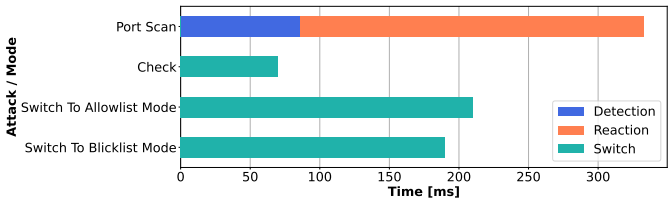

Fig. 3: Average Detection, Reaction, and Mode Switching Time on the IoT Devices in Milliseconds

– Preprint –
Accepted for publication at the SEAMS 2023. Final published version available at:
https://doi.org/10.1109/SEAMS59076.2023.00034

attacker's IP address was blocked via iptable rules, and their information was forwarded to the server. Further attempts by the attacker were unsuccessful, as the connection was blocked. In 99 subsequent runs, the attacker script took 2s per IoT device and could only scan two ports, before the recurring attacker IP address was banned 221ms after the first monitored port scan.

Sharing Attacker Information: We automatically forwarded the attacker's information from the IoT devices to the server, which took an average of 102ms. Our *Java Server Application* monitored the blocked IP event in 3ms, and the *CEP-Engine* analyzed the EPL constraints in 4ms. After two "blocked IP" events from different IoT devices, the constraint was violated, and actions were planned and executed within one millisecond. This involved running an *Ansible* playbook to add the attacker's IP address to the blocklist on all IoT devices, which took another 13s. Following this, further attacks on other IoT devices were denied.

IoT Device Offline Adaptation: We tested the offline adaptation by changing the server IP address so that IoT devices could not connect. Then we could retrace the switch to the more restrictive mode on all IoT devices and the adaption of the *F2b* configuration within 2s. After repeated attacks, the ban time increased exponentially. However, this adaptation increased the average processing time by 8ms. After switching back to the original server IP, IoT devices switched to the default mode in 2s. We also tested switching between *Blocklist-Mode* and *Allowlist-Mode* and back on IoT devices (500 runs), and found that it took an average of 206ms to switch to *Allowlist-Mode*, 195ms to switch back, and 74ms to check if the desired mode was already active (see Fig. 3). The RPi Zero took twice as long as the RPi 4, presumably due to limited hardware capabilities (RAM and CPU power).

Regarding RQ2 and DSEC4IOT's performance, we can conclude that we were able to automatically detect the attacks and respond in a timely manner. By sharing the attacker's information, we could protect all other IoT devices within 14s after the first attack. With adaptations, we could also protect offline devices, and mitigate ongoing attacks by exponentially increasing the ban time.

V. Discussion

Our *Self-protective IoT System* is not only built for a one-time task. With continuous *MAPE-K* loops on the server and the IoT device, it can adapt and improve over time. Initially, IT security professionals and operators may need to be involved in monitoring, analyzing, planning, and executing to avoid false alarms or overreactions. Using knowledge from past security events can improve staff productivity, efficiency, and effectiveness. As confidence in the system grows, suggested actions can be delegated to our DSEC4IOT framework.

Our test scenarios in Section IV are just examples and should be adjusted based on practical experience and the specific use case. Simply banning IP addresses is not a comprehensive solution to all security issues since attackers may manipulate log files. To prevent business interruptions and false positives, *F2b* allows configuring an ignore list for IP/DNS, such as private networks and company DNS hosts. Blocking specific countries is another way to mitigate attacks, especially for businesses operating in a specific country. Country IP zone files can be used to allow or deny specific IP addresses [36]. Additionally, setting rate limits for specific connection types and services may reduce DoS attacks.

VI. Threats to Validity

Like any other study, our work is subject to threats to validity [37]. Decentralization can increase security risks [38], such as a malicious insider manipulating all IoT devices connected to the server. However, our DSEC4IOT framework includes onboard protection components and allows individual devices to make adaptation decisions when the server is offline or compromised. For our evaluation, we conducted a case study in a research lab using realistic scenarios with commonly available hardware and open-source software. However, our approach may not be suitable for all environments or technologies. We plan to expand the scope and add vulnerability scenarios to demonstrate generalizability. Further testing, such as a user study, is needed to confirm usability.

VII. Related Work

As security issues can cause system faults and vice-versa, tactics and patterns for software robustness [39] are relevant for our framework. Self-protective IoT devices should be able to detect faults, recover from them, and prevent them in the future. That corresponds to the idea of the *MAPE-K* loop. Graceful degradation and reconfiguration, like mode switching, can help achieve goals, provide core functionality, and prevent system outages during an attack.

Witte et al. [40] propose the *Attack-Fault-Trees* modeling approach to analyze security issues, while Yuan et al. [41] provide a taxonomy, patterns, and tactics for self-protecting software systems, and propose the *Architecture-Based Self-Protection* approach [42] for web applications. The *Anomaly Behavior Analysis Intrusion Detection System* [43], [44] is used to detect known and unknown attacks in smart cyber infrastructures. While their system focuses on attack detection, our framework also considers attack reactions.

VIII. Conclusion

In this paper, we presented DSEC4IOT, a distributed *MAPE-K* framework for self-protective IoT devices that monitors and analyzes the device's environment to plan and execute actions for improving device and server security. The framework's knowledge base is updated for continuous adaptation and improvement, making IoT devices and servers more resilient to potential threats. Future work could use *Machine Learning* to detect abnormal behavior and leverage edge/fog computing for better performance and robustness.

Acknowledgments

This work has partially been supported by the LIT Secure and Correct Systems Lab and Linz Institute of Technology (LIT-2019-7-INC-316), funded by the State of Upper Austria.

– Preprint –
Accepted for publication at the SEAMS 2023. Final published version available at:
https://doi.org/10.1109/SEAMS59076.2023.00034

References

[1] Transforma Insights, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030, by use case," Statista, July 2022, Accessed: 2022-Dec-19. [Online]. Available: https://www.statista.com/statistics/1194701/iot-connected-devices-use-case

[2] Q. Wang, Y. Zhao, W. Wang, D. Minoli, K. Sohraby, H. Zhu, and B. Occhiogrosso, "Multimedia IoT systems and applications," in *2017 Global Internet of Things Summit*. IEEE, 2017, pp. 1–6.

[3] A. Ghasempour, "Internet of Things in Smart Grid: Architecture, Applications, Services, Key Technologies, and Challenges ," *Inventions*, vol. 4, no. 1, p. 22, 2019.

[4] M. Yun and B. Yuxin, "Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid," in *2010 Int'l Conf. on Advances in Energy Engineering*. IEEE, 2010, pp. 69–72.

[5] Y. U. Devi and M. Rukmini, "Iot in connected vehicles: Challenges and issues—a review," in *2016 Int'l Conf. on Signal Processing, Communication, Power and Embedded System*. IEEE, 2016, pp. 1864–1867.

[6] Json & Partners, "Size of the global market for Industrial Internet of Things (IIoT) platforms and apps for manufacturing industries from 2017 to 2021, with a forecast through 2025 (in billion U.S. dollars)," Statista, 2022, Accessed: 2022-Dec-19. [Online]. Available: https://www.statista.com/statistics/1339894/global-industrial-internet-of-things-platforms-and-apps-market-size/

[7] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," Cisco IBSG, April 2011, Accessed: 2022-Dec-19. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[8] D. Airehrour, J. Gutierrez, and S. K. Ray, "Secure routing for internet of things: A survey," *Journal of Network and Computer Applications*, vol. 66, pp. 198–213, 2016.

[9] C. C. Sobin, "A Survey on Architecture, Protocols and Challenges in IoT," *Wireless Pers. Commun.*, vol. 112, no. 3, p. 1383–1429, Jun 2020.

[10] D. Malani, J. Modi, S. Lilani, Y. Desai, and R. Dhanare, "Intrusion Detection Systems for Distributed Environment," in *2021 Third Int'l Conf. on Intelligent Communication Technologies and Virtual Mobile Networks*, 2021, pp. 98–103.

[11] M. Barrett, *Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1*. Gaithersburg, MD: US National Institute of Standards and Technology (NIST), Apr 2018, no. NIST CSWP 04162018.

[12] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[13] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation," in *2015 IEEE/ACM 10th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems*, 2015, pp. 13–23.

[14] D. Weyns, I. Gerostathopoulos, N. Abbas, J. Andersson, S. Biffl, P. Brada, T. Bures, A. Di Salle, P. Lago, A. Musil, J. Musil, and P. Pelliccione, "Preliminary results of a survey on the use of self-adaptation in industry," in *2022 Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems*. New York, NY, USA: Association for Computing Machinery, Aug 2022, p. 70–76.

[15] S. White, J. Hanson, I. Whalley, D. Chess, and J. Kephart, "An architectural approach to autonomic computing," in *Int'l Conf. on Autonomic Computing, 2004. Proc.*, May 2004, p. 2–9.

[16] H. Psaier and S. Dustdar, "A survey on self-healing systems: approaches and systems," *Computing*, vol. 91, pp. 43–73, 2011.

[17] 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*. Springer Berlin Heidelberg, 2013, pp. 76–107.

[18] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT systems: Design challenges and opportunities," in *2014 IEEE/ACM Int'l Conf. on Computer-Aided Design*, 2014, pp. 417–423.

[19] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[20] M. M. Shurman, R. M. Khrais, and A. A. Yateem, "IoT denial-of-service attack detection and prevention using hybrid IDS," in *2019 Int'l Arab Conf. on Information Technology*, 2019, pp. 252–254.

[21] A. K. Goel, A. Rose, J. Gaur, and B. Bhushan, "Attacks, countermeasures and security paradigms in IoT," in *2019 2nd Int'l Conf. on Intelligent Computing, Instrumentation and Control Technologies*, vol. 1, 2019, pp. 875–880.

[22] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," *Telecommunication systems*, vol. 73, no. 1, pp. 3–25, 2020.

[23] M. Riegler, J. Sametinger, M. Vierhauser, and M. Wimmer, "A Model-based Mode-Switching Framework based on Security Vulnerability Scores," *Journal of Systems and Software*, vol. 200, 2023.

[24] M. Riegler, J. Sametinger, and C. Schönegger, "Mode Switching for Secure Edge Devices," in *Database and Expert Systems Applications - DEXA 2022 Workshops*, G. Kotsis et al., Ed. Cham: Springer Int'l Publishing, 2022, p. 347–356.

[25] IBM, "An architectural blueprint for autonomic computing," Jun 2005, Third Edition. Accessed: 2022-Dec-19. [Online]. Available: https://www-03.ibm.com/autonomic/pdfs/ACBlueprintWhitePaperV7.pdf

[26] NVD, "National Vulnerability Database," Accessed: 2022-Dec-20. [Online]. Available: https://nvd.nist.gov

[27] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, p. 131, Dec 2008.

[28] J. Deogirikar and A. Vidhate, "Security attacks in iot: A survey," in *2017 Int'l Conf. on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Feb 2017, p. 32–37.

[29] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Jin, "Internet-of-Things Security and Vulnerabilities: Taxonomy, Challenges, and Practice," *Journal of Hardware and Systems Security*, vol. 2, no. 2, p. 97–110, Jun 2018.

[30] Nmap.org, "Nmap: Network Security Scanner," Accessed: 2022-Dec-2. [Online]. Available: https://nmap.org

[31] H. Sanghvi and M. Dahiya, "Cyber reconnaissance: an alarm before cyber attack," *Int'l Journal of Computer Appl.*, vol. 63, no. 6, 2013.

[32] Eclipse Foundation, "Mosquitto MQTT Broker," Accessed: 2022-Dec-2. [Online]. Available: https://mosquitto.org

[33] EsperTech, "Esper - Complex Event Processing & Stream Analytics," Accessed: 2022-Dec-2. [Online]. Available: https://www.espertech.com

[34] Red Hat, "Ansible," Accessed: 2022-Dec-2. [Online]. Available: https://www.ansible.com

[35] Fail2Ban, "fail2ban," Accessed: 2022-Dec-2. [Online]. Available: https://www.fail2ban.org

[36] IPdeny Project, "Ipdeny country block downloads," Accessed: 2022-Dec-3. [Online]. Available: https://www.ipdeny.com/ipblocks/

[37] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Berlin, Heidelberg, 2012.

[38] F. Quin, D. Weyns, and O. Gheibi, "Decentralized Self-Adaptive Systems: A Mapping Study," in *2021 Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems*, 2021, p. 18–29.

[39] R. Kazman, P. Bianco, S. Echeverría, and J. Ivers, *Robustness*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2022, no. CMU/SEI-2022-TR-004.

[40] T. Witte, R. Groner, A. Raschke, M. Tichy, I. Pekaric, and M. Felderer, "Towards Model Co-evolution Across Self-Adaptation Steps for Combined Safety and Security Analysis," in *2022 Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems*, 2022, pp. 106–112.

[41] E. Yuan, N. Esfahani, and S. Malek, "A Systematic Survey of Self-Protecting Software Systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 8, no. 4, pp. 17:1–17:41, Jan 2014.

[42] E. Yuan, S. Malek, B. Schmerl, D. Garlan, and J. Gennari, "Architecture-based self-protecting software systems," in *Proc. of the 9th Int'l ACM Sigsoft Conf. on Quality of Software Architectures*. New York, NY, USA: Association for Computing Machinery, Jun 2013, p. 33–42.

[43] J. Pacheco and S. Hariri, "IoT Security Framework for Smart Cyber Infrastructures," in *Proc. of the 2016 IEEE 1st Int'l Workshops on Foundations and Applications of Self* Systems*, 2016, pp. 242–247.

[44] J. Pacheco, X. Zhu, Y. Badr, and S. Hariri, "Enabling Risk Management for Smart Infrastructures with an Anomaly Behavior Analysis Intrusion Detection System," in *2017 IEEE 2nd Int'l Workshops on Foundations and Applications of Self* Systems*, 2017, pp. 324–328.