A Self-Configuration and Healing Controller To Analyze Misconfigurations of Clusters and IoT Edge Devices

Areeg Samir
UiT - The Arctic University of Norway
Tromsø, Norway
areeg.s.elgazazz@uit.no

Håvard Dagenborg
UiT - The Arctic University of Norway
Tromsø, Norway
havard.dagenborg@uit.no

ABSTRACT

The complex computational environments we often find running modern online services and IoT applications are vulnerable to security breaches and information leakage due to misconfigurations. In this paper, we propose a self-configurable and healing controller that detects, identifies, and recovers from various misconfigurations in complex computing environments that can span various backend clusters and edge components. We measure the accuracy of the proposed controller in terms of performance and reliability.

CCS CONCEPTS

• **Computer systems organization** → *Real-time systems*; • **Security and privacy** → **Security services**; *Software and application security*; **Systems security**.

KEYWORDS

Misconfigurations, Monitor, Detection, Identification, Recovery, Performance, IoTs, Clusters, HHMM, Markov Decision Process

ACM Reference Format:
Areeg Samir and Håvard Dagenborg. 2023. A Self-Configuration and Healing Controller To Analyze Misconfigurations of Clusters and IoT Edge Devices. In *Proceedings of 10th ACM Celebration of Women in Computing: womENcourage™ 2023 (womENcourage).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Misconfiguration might emerge when essential security settings of edge devices and system cluster parameters are either not implemented or implemented with errors such as leaving the default configuration settings unchanged or other technical issues [13]. Misconfiguration in edge devices and system clusters can impact the workload and the data flows within and across systems, which might lead to security breaches and performance degradation. For instance, if an edge device runs with default privileges, flaws in any system's component can be accidental (e.g., remote SSH open) or intentional (e.g., backdoor in component). Misconfigurations in core Kubernetes components (e.g., API server, Kubelet, Kube-proxy) result in compromizing all the nodes managed by that cluster. The management of configuration has been explored in the literature [1], [2], [3], [4], [5], [6], [11], [12]. However, more work is needed to correlate the misconfiguration of edge devices and system clusters to its observed performance degradation, identify its reasons, and recover the identified misconfiguration for optimizing the system's performance and reliability.

womENcourage, September 20–22, 2023, Trondheim, Norway
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 SYSTEM MODEL

Our system comprises hierarchical components with different configurations, resources, and policies. The components include gateways, sensors, services, edge devices, clusters, nodes, containers, and healthcare participants, including their roles and access control.

Misconfiguration can occur at different levels. For instance, at the edge device level, misconfiguration (e.g., missing authentication and authorization [8]) could affect the scores of the Medtronic heart-rate monitor, allowing a nearby attacker to change the settings of a patient's device. At the cluster level, misconfigurations (e.g., vulnerable product version [7], [9], no parameters validation [10]) might allow an attacker to gain root-level access on the host and exploit system processes. We focus on common misconfigurations (errors) in Kubernetes, Azure, and Docker Swarm that negatively impact system workload (fault) and cause performance degradation (failure) on performance metrics [13].

3 MISCONFIGURATIONS MANAGEMENT

First, we analyzed various misconfigurations by classifying misconfigurations under three categories (e.g., IoT edge, application, and node or cluster) [13]. Then, we checked the workload for the components by estimating the dissociation between the failures and the faults [15]. After that, we checked the configuration settings against the benchmarks of Azure Security, CIS Docker, and Kubernetes. For any mismatch between the settings and the requirements of secure deployment in components, the controller.

We controlled the access to the system services by controlling the information flow from/to the system and managing the interaction of the healthcare participants. We extended our HHMM model [13], which tracks the path and dependency of misconfigurations in edge devices and system clusters, with a set of controlling labels comprising tags, each representing a specific integrity issue and outlining the information flow permitted. We defined a role-based access space and a set of actions policy for every participant to allow specific participants access to specific system services. Once access to the system is secured, the controller moves to the detection phase.

At the detection phase, we graphically represented our system under observation to map the hidden misconfiguration settings (error) from the observer to the performance metric (failure). We choose HHMM as every component, and its configuration settings dependency can be represented as a hierarchical interlinked HMM. We compared the detected hierarchies against the observed ones to identify the misconfigured state type and its impact on the flow of information. The detected path with the lowest probabilities was considered vulnerable to misconfigured states. For each misconfigured state, the configuration settings (key-value pairs) were

Table 1: Detection Evaluation

Models	Recall	Accuracy
HHMM	95.01%	94%
CRFs	92.86%	92%

extracted from the manifest based on the Kind and API version objects to be checked against centrally managed correct configuration settings stored in Knowledge storage. We iterated through the manifests settings to check each key-value pair. We computed the confidence score by considering the key type (p-value$\leqslant$0.05) to validate our hypothesis against the difference between the manifests. A low confidence score indicates a difference between the configuration obtained from the anomalous state (actual state) and the correct configuration (desired state), representing the improperly configured state. The result of this phase is stored in Knowledge storage to identify the type of misconfiguration and threat (if any) [13].

Based on the misconfigured key-value pair and the Kind object obtained from the anomalous state, we initialized our model states and observations parameters to represent the misconfigured setting that belongs to the detected misconfigured state. For each state, we checked the misconfiguration type hidden from the observer considering the category of the state, and we calculated the probability distribution considering the system's overall performance. The misconfiguration with the lower performance is discarded, and the one with higher performance is kept. In case the misconfiguration is not defined within the cases, the controller records the new characteristics of the misconfiguration type and assigns unique identifiers to the selected items [13].

For each misconfigured state type, the controller selects an optimal recovery policy with proper recovery action(s) and observes the environment's performance after applying the action. Here, the controller receives a reward that refers to the performance enhancement for the observed monitoring metrics. If the action is applied successfully, the controller marks the component as recovered and keeps a profile for the applied actions in the knowledge. Otherwise, the controller applies another action. Based on the applied actions, observations, and rewards obtained from the applicable actions, the controller continuously updates the recovery policy to find an optimal policy that maximizes the expected cumulative long-term reward received during recovery. The controller stops the policy update once the difference between the update becomes marginal [14].

4 RESULTS AND DISCUSSION

The controller is trained on some of the identified misconfigurations that allow privilege escalation to the host [8], [9]. We used different values of observation sequence length (n = 30, 60, 90, 120, 150). We evaluated the detection performance by comparing the HHMM with Conditional Random Fields (CRFs) and measuring Recall and Accuracy. The results show that the performance of the proposed detection is better than the CRF, as it correctly identified true positives of misconfiguration with 95% recall and 94% accuracy; see Table 1. We observed that when the length of observation sequences was 60, a detection rate of around 88% could be obtained with zero false alarms. In contrast, if the length of observation sequences equals 140, approximately 97% detection rate can be obtained.

The controller performance was almost the same for misconfiguration types, with a minor recovery time deviation of around 100 seconds. The deviation returned to the correlation with the failure in the system. We solved that by utilizing the sequence of failures occurrence to correlate failures that share the same observations to a unique fault and to handle the failure based on its precede occurrence. Nevertheless, anomalous behavior [10] that is not covered by the training set could be found in the test set. To overcome that, we increased the training dataset size, which participated in enhancing the controller performance.

To measure the successfully recovered components to the total number of misconfigured components, we measured the controller Average Overall Rate (AOR). After multiple runs, the average rate was around 97.66%, which means that the recovery could not handle a small number of misconfigurations, though the unhandled anomalous behavior decreased dramatically with more training data. The results stated that the controller performed better with increased training dataset size and observations.

ACKNOWLEDGMENT

This research was funded in part by The Research Council of Norway under grant number 274451 and 263248.

REFERENCES

[1] Tatsuhiro Chiba, Rina Nakazawa, Hiroshi Horii, Sahil Suneja, and Seetharami Seelam. 2019. ConfAdvisor: A Performance-Centric Configuration Tuning Framework for Containers on Kubernetes. *IEEE International Conference on Cloud Engineering IC2E*, 168–178.

[2] Shuvalaxmi Dass and Akbar Siami Namin. 2021. Reinforcement Learning for Generating Secure Configurations. *Electronics 2021, Vol. 10, Page 2392* 10 (9 2021), 2392. Issue 19. https://doi.org/10.3390/ELECTRONICS10192392

[3] Fairwinds. 2023. Kubernetes Benchmark Report Security, Cost, and Reliability Workload Results. https://www.fairwinds.com/kubernetes-config-benchmark-report

[4] Michael Hicks, Stephen Tse, Boniface Hicks, and Steve Zdancewic. 2005. Dynamic Updating of Information-Flow Policies. *Proceedings of the International Workshop on Foundations of Computer Security FCS*, 7–18.

[5] Alessandro Mascellino. 2022. Nearly One Million Exposed Misconfigured Kubernetes Instances Could Cause Breaches. https://www.infosecurity-magazine.com/news/misconfigured-kubernetes-exposed/

[6] Shana Moothedath, Dinuka Sahabandu, Joey Allen, Andrew Clark, Linda Bushnell, Wenke Lee, and Radha Poovendran. 2020. Dynamic Information Flow Tracking for Detection of Advanced Persistent Threats: A Stochastic Game Approach. *arXiv:2006.12327* (6 2020). http://arxiv.org/abs/2006.12327

[7] NVD. 2019. CVE-2019-5736. https://nvd.nist.gov/vuln/detail/CVE-2019-5736

[8] NVD. 2019. CVE-2019-6538. https://nvd.nist.gov/vuln/detail/CVE-2019-6538

[9] NVD. 2020. CVE-2020-10749. https://nvd.nist.gov/vuln/detail/cve-2020-10749

[10] NVD. 2022. CVE-2022-0811. https://nvd.nist.gov/vuln/detail/cve-2022-0811

[11] Alif Akbar Pranata, Olivier Barais, Johann Bourcier, and Ludovic Noirie. 2020. Misconfiguration discovery with principal component analysis for cloud-native services. *Proceedings - 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing, UCC 2020*, 269–278. https://doi.org/10.1109/UCC48980.2020.00045

[12] Akond Rahman, Shazibul Islam Shamim, Dibyendu Brinto Bose, and Rahul Pandita. 2023. Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study. *ACM Transactions on Software Engineering and Methodology* (1 2023), 1–37. https://doi.org/10.1145/3579639

[13] Areeg Samir and Håvard Dagenborg. 2023. A Self-Configuration Controller To Detect, Identify, and Recover Misconfiguration At IoT Edge Devices and Containerized Cluster System. *The 9th International Conference on Information Systems Security and Privacy ICISSP*, 765–773. https://doi.org/10.5220/0011893700003405

[14] Areeg Samir and Håvard Dagenborg. 2023. Self-Healing Misconfiguration of Cloud-Based IoT Systems Using Markov Decision Processes. *The 13th International Conference on Cloud Computing and Services Science CLOSER*, 244–252.

[15] Areeg Samir and Claus Pahl. 2019. A Controller Architecture For Anomaly Detection, Root Cause Analysis and Self-Adaptation for Cluster Architectures. *The Eleventh International Conference on Adaptive and Self-Adaptive Systems and Applications ADAPTIVE*, 75–83.