CubeMig: MTD Live Migration in Kubernetes with LLM-Augmented Post-Incident Analysis

Michael Meier Azhari*, Wissem Soussi*§, and Gürkan Gür*
*Zurich University of Applied Sciences (ZHAW), Switzerland
§University of Zurich (UZH), Switzerland
E-mails: meierm78@students.zhaw.ch, {sous, gueu}@zhaw.ch

***Abstract*—Future networks are expected to rely heavily on cloud-native technologies. However, the security and resilience of those systems deserve more attention. CubeMig presents an approach to enhancing security in Kubernetes environments, enabling pods to live migrate, leveraged as part of a Moving Target Defense (MTD) strategy. CubeMig showcases reactive defensive mechanisms by incorporating automated live migrations as a response to threats detected in near real-time at the OS kernel level, using an eBPF-based approach. The mitigation process is further enhanced with forensic analysis on the checkpoint of the migrated container, providing insights into the compromised containers offline and instantiating the container in a sandboxed environment for further online analysis. Finally, we augment the forensic analysis output using LLMs, generating human-explainable analysis from the forensic logs to support post-incident investigation and providing relevant security recommendations. Experimental results validate the approach's effectiveness in a human-in-the-loop setting, showcasing the system's ability to detect and respond to attack scenarios such as reverse shell execution, log tampering, and system destruction.**

I. Introduction

Containerization has transformed the deployment and management of applications by providing portability, scalability, and resource efficiency [1]. It is also touted as a fundamental technology for future networks like 6G with the integration of cloud-native technologies and deployment of Containerized Network Functions (CNFs). As a leading container orchestration platform, Kubernetes enables seamless management of containerized applications across clusters. In modern cybersecurity, Moving Target Defense (MTD) has emerged as a powerful strategy to enhance system security [2]. MTD focuses on dynamically changing the attack surface, making it more difficult for attackers to predict and exploit vulnerabilities. Within MTD, container live migration can serve as one of the approaches to achieve this dynamic shift by relocating workloads, disrupting potential attack vectors [3]. Additionally, MTD via live migration can be useful for isolating an infected container by moving it to a secure cluster, allowing deeper analysis when an unknown attack occurs. This isolation helps contain the breach, preventing the infection of other applications and the unauthorized exfiltration of data.

This paper presents CubeMig, a proof-of-concept (PoC) framework relying on open-source components to live-migrate containers running in Kubernetes environments without disrupting the high-level orchestration control. This enables native multi-cluster live migration for enhanced container fault tolerance, physical isolation, and forensic analysis. We then extend the tool by integrating attack scenarios to improve security capabilities. This work makes three primary contributions. First, it integrates near real-time threat detection through Falco [4], developing an eBPF-based reactive live migration to evade security attacks and facilitate forensic analysis of static and dynamic container runtimes. Second, it introduces forensic analysis using `checkpointctl`, analyzing processes, memory usage, and network connections within migrated containers. Third, it leverages Large Language Models (LLMs) to offer AI-based recommendations, helping administrators to interpret system logs quickly, identify the trigger that activated the detection system, and address potential vulnerabilities. CubeMig also delivers a user-friendly Angular-based front-end for configuring, visualizing, and managing migrations, making the system accessible to stakeholders and interested parties. The open-source code is available on GitHub[1].

This paper details CubeMig's methodology, implementation, and results, discussing challenges faced and outlining potential avenues for future improvements. In Section II, we provide relevant technical background. Then we present the system architecture in Section III. We elaborate on experimental results and key takeaways in Section IV, while we outline technical challenges and limitations of our work in the following section. Finally, we conclude with discussions and future work in Section VI.

II. Technical Background and Related Work

MTD enhances security by dynamically changing a system's configuration, going beyond static defenses like firewalls and encryption [5]. MTD techniques are broadly categorized as shuffling, diversity, and redundancy. In 5G and prospective 6G networks, this can involve moving virtual network functions (VNFs/CNFs) between cloud infrastructures or using Software-Defined Networking (SDN) to alter network interfaces and paths [6]–[8]. While prior work has applied MTD in various domains, its use in current 5G NFV environments is less explored. Some studies have used SDN/NFV for MTD to thwart DDoS attacks [9] or to attach security functions dynamically [10], [11]. However, MTD strategies that directly manipulate VNFs/CNFs via actions like live migration remain an open area for research.

[1]CubeMig: https://github.com/wsoussi/CubeMig

 DOI: 10.1109/NFV-SDN66355.2025.11349649

A. Container Orchestration and Migration

Container orchestration is the process of managing the lifecycle of containers, including deployment, scaling, and operation, across distributed systems. Kubernetes has emerged as an industry standard for container orchestration, enabling efficient management of applications in clustered environments [12]. Kubernetes has recently introduced the Kubelet Checkpoint API, a feature that facilitates the checkpointing and restoration of container states [13]. As of the writing of this paper, the Kubelet Checkpoint API is in its beta phase (`Kubernetes v 1.30`), showing its low maturity level. The Kubelet Checkpoint API is the core of the migration process used in CubeMig. This is enabled by the open-source Checkpoint/Restore In Userspace (CRIU) library [14]. CRIU allows the low-level container runtime to capture the state of a container, including memory, processes, and network connections, known as a checkpoint. The checkpoint can then be transferred and restored on a different node or cluster, allowing migration. Container live migration is connected to, among other tasks, MTD operations used to disrupt attackers by dynamically altering the system's attack surface, relocating workloads to secure environments when threats are detected [15].

B. Real-time security event detection

Falco, an open-source cloud native security tool, is used for near real-time attack detection [4]. It monitors system calls and kernel-level events to identify behaviors that deviate from predefined security rules, using an eBPF probe or a kernel module. Examples of such behaviors include unauthorized network access, file tampering, and attempts to escalate privileges. In this work, Falco is deployed in its out-of-the-box configuration without any rule extensions, demonstrating its baseline capabilities. Upon detecting an anomaly in the system, Falco can be configured to send an output to another system.

C. Post-Incident Analysis

To gain a deeper understanding of what happened in a migrated container, a forensic analysis can be performed after a migration is triggered. This analysis provides detailed insights into the state of the container at the time the checkpoint is created. CubeMig uses `checkpointctl`, a tool designed to analyze container checkpoints created during migration. The forensic analysis includes critical data such as:

- The processes running inside the container
- Memory consumption for each process
- Active TCP connections within the container
- Changes made to the container compared to its base image

This information is valuable for understanding the nature of the attack and identifying potential vulnerabilities in the containerized environment.

D. AI-Based Recommendations for Mitigation

In addition to detecting and analyzing attacks, CubeMig integrates an LLM to generate automated security recommendations based on forensic data. When an attack is detected and a migration is triggered, the `checkpointctl` analysis of the checkpointed container state are compiled into a structured format and sent to an LLM. By parsing the container's forensic data, the LLM offers suggestions regarding potential vulnerabilities and remediation steps. For instance, it may highlight suspicious processes, TCP connections, or new files. The goal is to assist administrators in their post-incident investigations, providing initial insights more quickly than would be possible through purely manual analysis.

This feature is configurable within the system's rules and can be enabled or disabled based on the user's needs. Although LLM-driven suggestions can accelerate the incident response process, it is important to note that the model's accuracy and contextual understanding may not always be accurate. Despite these limitations, AI-based suggestions represent a promising direction for enhancing and speeding up security incident response in containerized environments within a human-in-the-loop setting.

III. System Architecture

The architecture of CubeMig integrates attack detection, automated container migration, forensic analysis, and a user-friendly front-end interface, all built around a Kubernetes-based environment. The testbed consists of two Kubernetes clusters, a centralized virtual machine (VM) for managing migrations, and several supporting components, as depicted in Figure 1. The two test clusters are used to integrate and evaluate CubeMig's migration workflow. These are:

- **Primary Cluster (cluster 1):** This cluster simulates a production environment and contains a vulnerable application

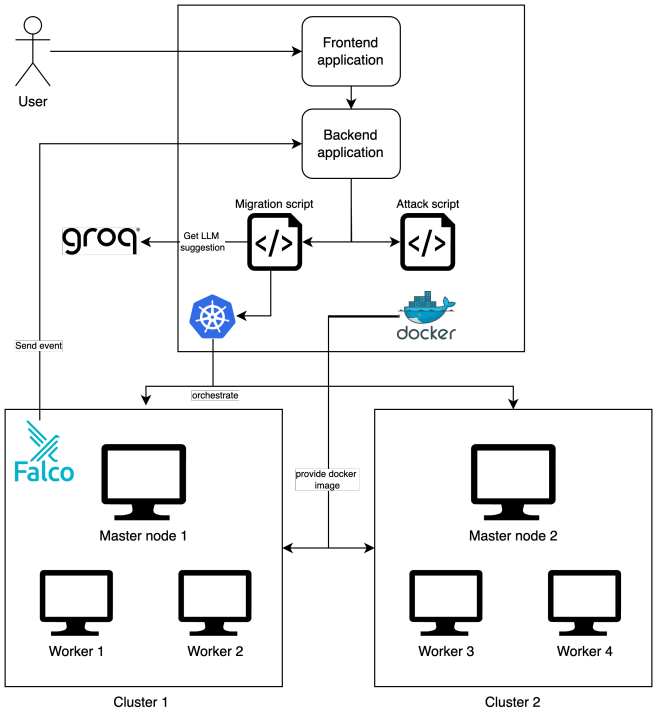

Fig. 1. System architecture and testbed

used to simulate attacks. It serves as the source environment where Falco is deployed to monitor security events, which will initiate migrations and security actions.
- **Secure Cluster (cluster 2):** This cluster acts as the target for migrations, providing a safer environment for workloads after an attack is detected. It ensures that an exposed application is isolated from the production environment in a secure zone and can be monitored more carefully. This also protects other applications from being infected, in case the attacker manages to do a container escape attack. Such a cluster is configured not to have any outbound connections, ensuring that no connection to the outside world is possible.

A VM is connected to both clusters and hosts CubeMig, orchestrating the live migration process. The VM connects with NFS to the file storage systems of both clusters to retrieve the latest container checkpoints, enabling faster migrations. Additionally, the VM hosts a local Docker registry for storing container images created during the migration process. Using a local Docker registry ensures that a migration can be executed without having to fetch new images from the Internet remotely. It also deploys the front- and back-end CubeMig applications, providing an interface for monitoring and managing the migration workflow and attack simulation.

A. CubeMig architecture

CubeMig, developed using FastAPI, acts as the decision-making engine and orchestrator of the system. It serves multiple purposes as mentioned below.

*1) **Event Collector**:* CubeMig alerts endpoint serves the purpose of catching Falco events. This endpoint is configured as an HTTP output in Falco configuration. With this configuration, Falco will send any event to CubeMig. Cubemig then processes the received event. Based on the configuration file, it either migrates the container, logs the event, or ignores the event.

*2) **Kubernetes extension**:* CubeMig uses the Kubernetes SDK to show Kubernetes information in the front-end. It provides two Kubernetes endpoints:

- GET pods endpoint: Return all pods running in a cluster. The information returned includes the pod name, status, and how long it has been running. The information of which cluster to show is provided by the user as a query parameter.
- DELETE pod endpoint: Besides showing pod information, the endpoint also allows deletion of the Kubernetes pod, allowing orchestration directly from the front-end application.

*3) **Manual migration**:* Triggering a manual migration is also possible in CubeMig. With its migrate endpoint, a container can be migrated from a source cluster to a target cluster. The user needs to provide the pod name, source cluster, and target cluster in the request. This endpoint will then call the migration function to initiate a migration.

*4) **Attack simulation**:* As we would like to evaluate the methodology using an attack simulation, CubeMig provides the endpoint for an attack simulation. The information required is the application name and the attack type. Currently, only data destruction, log tampering, and reverse shell are supported. Upon receiving a call to this endpoint, CubeMig will trigger the attack function for the specific application, which then initiates the attack simulation on the given application.

It is worth mentioning that the input of this endpoint is the application name, and not the pod name. This is intentional because the attack simulation tries to attack a running application, which does not care which pod is currently running. Two threats can be simulated in CubeMig, exploiting known existing vulnerabilities listed in Table III. The first one exploits a vulnerability in the Spring Boot Java framework, enumerated by the Common Vulnerability Enumeration (CVE) database as CVE-2022-22963. It is executed in a running container by passing an encrypted header to a request of the Spring Boot application, enabling remote code execution (RCE). Following this exploit, three attacks can be selected to be executed: reverse shell, data destruction, and log tampering. The second one targets Redis-based applications vulnerable to CVE-2022-0543. Using this vulnerability, it is also possible to trigger RCE and gain control of the container. The attack is possible due to a vulnerability in the Lua sandbox.

TABLE I
VULNERABILITIES AND ATTACKS SIMULATED BY CUBEMIG.

Vulnerability (CVE)	Target	Simulated Attack
CVE-2022-22963	Spring Boot	Reverse shell
	Spring Boot	Data destruction
	Spring Boot	Log tampering
CVE-2022-0543	Redis	Remote code execution

*5) **Configuration editor**:* To allow users to have control over which Falco event should trigger a migration, CubeMig provides also an endpoint to add and remove the migration configuration dynamically. This endpoint will then edit the JSON configuration file, which is then read by the event collector function of CubeMig.

*6) **Log viewer**:* The log viewer allows the front-end application to display the log file as a tree structure. It also allows user to view and download a specific log file.

B. Migration Function

The migration function, hosted on the VM, facilitates the live migration process using CRIU. It is based on a previous work [3] of the authors. CubeMig extended the function to include:

- Forensic analysis using `checkpointctl`, which provides detailed insights into container states.
- AI-based suggestions by sending forensic data to an LLM for security recommendations.
- Performance analysis and logging to ensure detailed tracking of migrations.

CubeMig connects to both clusters to manage container checkpoints and uploads the resulting container images to the local Docker registry.

C. GroqCloud

Another critical component of CubeMig is the integration of LLM into the migration process. The AI-based suggestion is enabled using GroqCloud [16]. GroqCloud is a platform created to access AI LLM models. It provides an OpenAI-like API, which a registered user can access using a private API key. By the time this paper was written, GroqCloud provided access to various models, including Llama 3.3, OpenAI Whisper, and Deepseek R1.

In CubeMig, the GroqCloud API is called after a migration has been successfully conducted. The generated forensic analysis from `checkpointctl` is passed as user input. Additional prompting (instruction) was needed to allow the LLM model to process the given forensic analysis. This instruction is passed on each API call as a system input.

IV. Experiments and Key Takeaways

This section presents the experiments conducted to evaluate the system components and the corresponding results. The experiments include testing the migration workflow, attack detection performance, forensic analysis capabilities, and AI-generated suggestions. The evaluation is mainly qualitative but provides insights into the value of CubeMig for security management and automation.

A. Attack scenarios

A variety of attack scenarios were evaluated to demonstrate how container migration can mitigate threats in near real-time. However, as noted above, Falco's default ruleset does not detect all possible threats, limiting the scope of our simulations. For instance, certain ransomware or data exfiltration techniques might not be flagged with the default Falco rules.

To conduct these experiments efficiently, we employed the Atomic Red Team tool [17], which was deployed as a container in Cluster 1. This tool offers a comprehensive library of attack techniques, ranging from privilege escalation to lateral movement. Automating these attacks enabled us to test multiple scenarios simultaneously, alleviating the experimentation process. Through this systematic approach, we observed that CubeMig successfully detected and migrated certain compromised containers when Falco's built-in rules were triggered. Atomic Red Team is used to implement the three attack simulations on a Spring Boot application exploiting the vulnerability CVE-2022-22963: data destruction, log tampering, and reverse shell.

B. Migration Results

Three attack scenarios were conducted to evaluate the container migration process and assess its security impact. The following part details how migration mitigates each type of attack and the observed outcomes.

1) Data Destruction: This attack targets the container at the root level, attempting to wipe out all data and thereby compromising the container's availability. Without a migration mechanism, a successful data wipe would make the container unusable. However, when migration is in place, several security measures take effect. Falco detects bulk deletion activities by identifying commands such as `shred`, `mkfs`, and `mke2fs` executed by an unauthorized user. It then notifies the microservice, which immediately creates a checkpoint and triggers the migration. After a successful migration, a new container instance is launched to ensure continuous availability. During testing, only a small number of files were deleted before migration completed, preventing the container from being completely compromised. This demonstrates that while migration cannot fully stop a data destruction attack, it can significantly mitigate the impact and help preserve critical data.

2) Log Tampering: This scenario simulates an attacker modifying log data within a running container in an effort to conceal unauthorized activities and remain undetected. The integrity of audit logs is crucial for both real-time monitoring and post-incident forensic analysis. By migrating the container to a secure environment, the attack is contained. Falco detects potential tampering of system logs and generates an alert to the microservice. Upon receiving the alert, the microservice initiates the process to create and migrate the checkpoint. Once the migration succeeds, the attacker loses access to the system. The forensic examination then identifies which files were altered, providing insight into the attacker's actions and timeline. Although some log files were already corrupted by the time migration took effect, the damage was confined to a smaller scope. The migration process thus prevents further tampering while preserving enough evidence for investigation.

3) Reverse Shell: In this attack, an interactive Bash session is launched on the container and connected back to a remote host, establishing a TCP connection to the attacker's machine. Falco detects the unauthorized redirection of standard input and output to a network socket. It then informs the microservice, which triggers the migration. The compromised container is migrated, closing any open TCP connections and cutting off the attacker's access. Tests show that the reverse shell connection can be terminated in under five seconds, effectively blocking further malicious activity. Deleting the container immediately after creating a checkpoint could close the connection even faster, but this approach risks permanent data loss if the checkpoint is corrupted and the migration fails. Consequently, CubeMig opts to delete the container only after a successful migration, ensuring a safer balance between rapid response and data protection.

C. Attack Detection Results

This evaluation extensively tested Falco's detection capabilities. In CubeMig, only Falco's default rule set was used, allowing for effective identification of basic attacks and immediate event notifications to listeners. However, more sophisticated threats remained undetected, showing the need to customize Falco's rules to suit specific application requirements. Overly restrictive rules can also create risks by generating excessive alerts (high false positive rate) and triggering unnecessary actions.

The test scenarios show the following outcomes:

- Falco reliably detected reverse shell attacks, unauthorized file access, and log tampering.
- Alerts were forwarded to the microservice without delay, enabling prompt migration triggers.
- Ransomware encryption attacks went undetected due to missing rules, emphasizing the importance of rule customization for broader coverage.

D. Forensic Analysis Results

Immediately after an attack, details on open TCP connections, running processes, and modified files proved especially useful for understanding the nature and scope of the intrusion. However, testing revealed that not all deleted files were correctly reported in the list of changed files, indicating a need for further refinement. Supplementing automated checkpoint-based analysis with manual methods could provide a more comprehensive view of malicious activities.

In its current version, CubeMig focuses exclusively on checkpoint analysis. A potential improvement would involve examining the container once it continues operation in the secure cluster. This real-time inspection could help identify residual attack effects and clarify what the outcome might have been without migration.

E. AI Suggestions Output

The AI-based suggestion system was tested by sending forensic analysis data to an LLM. For CubeMig, `llama-3.3-70b-versatile` was used. To only test the model's ability to diagnose problems from forensic data alone, all references to the specific event that triggered the analysis were removed. Even without that context, the model identified potential issues with high accuracy and suggested useful remedial measures for system administrators. As shown in Table II, the model correctly identified 94% of reverse-shell and data-destruction attacks, and 54% of log-tampering cases. The lower detection rate for log tampering is explained by the simulation method: performing an RCE was required before modifying the logs. The migration log provided to the LLM contained socket information and a list of changed files, which allowed the model to detect log tampering. However, since the RCE represents a more severe and evident attack, it was better recognized by the model. This result shows that the LLM tends to prioritize higher-impact or more explicit attack patterns when multiple activities occur in sequence.

In addition to measuring detection accuracy, the clarity of the model's diagnostic outputs was evaluated through manual classification. Each LLM response was categorized as either strong or weak depending on how explicitly it described the detected attack type. Outputs labeled as strong clearly identified the attack in a direct and unambiguous manner, while weak outputs required further interpretation to understand the underlying issue. As shown in Table III, 80% of data-destruction detections, 77% of log-tampering detections, and 68% of reverse-shell detections were classified as strong. This indicates that, in most cases, the model not only recognized the attacks but also communicated its findings in a way that could assist human operators during incident analysis.

These findings demonstrate the potential of the model as a human-in-the-loop tool to assist analysts in identifying alerts and refining diagnostic hypotheses. Nonetheless, the quality of the AI-generated outputs strongly depends on the completeness and accuracy of the forensic data, highlighting the need for robust data collection practices. The model's recommendations are expected to improve further if contextual information, such as the Falco rule that triggered the migration, is included.

TABLE II
ACCURACY OF LLM SUGGESTIONS.

Simulated Attack	Number of trials	Correct prediction	Accuracy (%)
Reverse shell	50	47	94
Data destruction	50	47	94
Log tampering	50	28	54

TABLE III
CLARITY OF LLM DETECTION.

Simulated Attack	Correct predictions	Strong clarity (%)
Reverse shell	47	68.1
Data destruction	47	80.9
Log tampering	28	77.8

V. CHALLENGES AND LIMITATIONS

This section outlines the challenges encountered during the development and implementation of the system, as well as the limitations observed during testing.

A. Limited Detection Capabilities

The detection capabilities of the system are limited by the predefined rules available in Falco's out-of-the-box configuration. While basic attacks such as reverse shell and log tampering were detected successfully, more complex scenarios, such as ransomware data encryption, would necessitate specific detection mechanisms [18]. Customizing and extending Falco's rule set is essential to improve the system's ability to handle advanced and evolving threats. A separate research direction focuses on proactive MTD strategies, which initiate MTD operations prior to attack detection. However, the resource cost and potential service disruption of operations such as live migration create a fundamental trade-off. Consequently, a primary challenge in this domain is optimizing the security benefits of proactive MTD against its inherent performance overhead and resource consumption [19].

B. Operational Improvements

The system relies on an NFS to transfer checkpoint files between the clusters. However, periodic disconnections of the NFS service caused disruptions in the migration workflow.

These disconnections required manual reactivation of the service, resulting in increased operational overhead. Automating the setup and recovery process for the NFS connection is a crucial improvement to increase system reliability.

In the CubeMig application, the logging mechanism can be extended to include more granular information about system performance and attack scenarios, aiding in better diagnostics and debugging. Moreover, usability improvements such as advanced filtering and sorting options for logs and configuration management would improve the tool.

C. LLM Improvements

This project demonstrates the use of an LLM with a Human-in-the-Loop workflow. However, further refinement of the model's reasoning and output quality is needed to enhance system administrators' response time and diagnostic accuracy. One potential improvement is to include the triggered alert as part of the model's input, enabling the LLM to identify issues with greater precision. Another improvement involves structuring the model's responses in a predefined format, ensuring that recommendations are concise, consistent, and easier to interpret during incident analysis. Additionally, optimizing the structure and content of the input prompt to better align with the LLM's capabilities could further improve its detection accuracy and contextual understanding.

VI. Conclusion and Future Work

This paper explored the integration of attack scenarios into container migration workflows within Kubernetes environments for MTD and presented a PoC tool, namely CubeMig. CubeMig demonstrated the feasibility of combining security mechanisms with container orchestration to enhance resilience against cyber threats. Experimental results validated the system's ability to detect and mitigate common attack scenarios, such as reverse shell execution and log tampering, while maintaining minimal application downtime during migrations. However, challenges such as the limited detection capabilities of Falco without customizations were observed. There are several areas for further development and improvement to enhance the system's capabilities and robustness. Future work will focus on expanding attack scenarios, extending Falco's detection rules, automating NFS management, and refining the front-end application to increase the system's robustness and usability further.

Acknowledgment

This work was supported partially by (a) the University of Zürich UZH, Switzerland and (b) the Horizon Europe Program's project NATWORK, Grant Agreement No. 101139285 funded by the Swiss State Secretariat for Education, Research, and Innovation SERI, under Contract No. 22.00642.

References

[1] P. Sharma, L. Chaufournier, P. Shenoy, and Y. C. Tay, "Containers and virtual machines at scale: A comparative study," in *Proceedings of the 17th International Middleware Conference*, Middleware '16, (New York, NY, USA), Association for Computing Machinery, 2016.

[2] J. Zheng and A. S. Namin, "A survey on the moving target defense strategies: An architectural perspective," *Journal of Computer Science and Technology*, vol. 34, no. 1, pp. 207–233, 2019.

[3] R. Kolodziejczyk, A. Mamaril, W. Soussi, and G. Gür, "Containers on the move: An experimental analysis of container migration in kubernetes," in *ICC 2025 - IEEE International Conference on Communications*, pp. 1–7, 2025.

[4] Sysdig, "Falco." https://falco.org, 2025. Accessed: 2025-7-15.

[5] J.-H. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 709–745, 2020.

[6] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 127–132, 2012.

[7] N. Mayone, P. Kunz, B. Yigit, W. Soussi, B. Stiller, and G. Gür, "IPv6 connection shuffling for moving target defense (MTD) in SDN," in *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 373–378, IEEE, 2024.

[8] W. Soussi, M. Christopoulou, T. Anagnostopoulos, G. Gür, and B. Stiller, "Topofuzzer — a network topology fuzzer for moving target defense in the telco cloud," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–5, 2023.

[9] A. Aydeger, N. Saputro, and K. Akkaya, "A moving target defense and network forensics framework for ISP networks using SDN and NFV," *Future Generation Computer Systems*, vol. 94, 2019.

[10] M. Rawski, S. Kukliński, P. Sapiecha, M. Pelka, G. Przytula, P. Wojslaw, and K. Szczypiorski, "MMTD: MANO-based moving target defense for corporate networks," in *2020 World Conference on Computing and Communication Technologies (WCCCT)*, 2020.

[11] G. Chollon, D. Ayed, R. A. Garriga, A. M. Zarca, A. Skarmeta, M. Christopoulou, W. Soussi, G. Gür, and U. Herzog, "ETSI ZSM driven security management in future networks," in *2022 IEEE Future Networks World Forum (FNWF)*, pp. 334–339, 2022.

[12] The Kubernetes Authors, "Kubernetes description." https://kubernetes.io, 2025. Accessed: 2025-7-15.

[13] A. Mamaril, R. Kolodziejczyk, W. Soussi, and G. Gür, "Exploring live payload migrations for MTD in microservices architecture," in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, pp. 1–5, 2024.

[14] CRIU Project, "Criu: Checkpoint/restore in userspace." https://criu.org, 2025. Version 4.1.

[15] W. Soussi, G. Gür, and B. Stiller, "Democratizing container live migration for enhanced future networks - a survey," *ACM Comput. Surv.*, vol. 57, Dec. 2024.

[16] Groq, "Groqcloud." https://groq.com/groqcloud, 2025. Accessed: 2025-07-10.

[17] Atomic Read Team, "Atomic Red Team." https://www.atomicredteam.io/, 2025. Accessed: 2025-07-07.

[18] L. Fernández Maimó, A. Huertas Celdrán, A. L. Perales Gómez, F. J. García Clemente, J. Weimer, and I. Lee, "Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments," *Sensors*, vol. 19, no. 5, 2019.

[19] W. Soussi, G. Gür, and B. Stiller, "Moving target defense (MTD) for 6G edge-to-cloud continuum: A cognitive perspective," *IEEE Network*, vol. 39, no. 1, pp. 149–156, 2025.