# Exploratory Performance Evaluation of VM Migration as MQTT Moving Target Defense

Matheus Torquato*†, Tiago Cruz*, Denis Rosário‡, Michele Nogueira§, Eduardo Cerqueira‡
*University of Coimbra, CISUC/LASI, DEI, Coimbra, Portugal
†Federal Institute of Alagoas, Campus Arapiraca, Arapiraca, Brazil
‡Federal University of Pará, Belém, Brazil
§Department of Computer Science - Federal University of Minas Gerais, Brazil
{mtorquato, tjcruz}@dei.uc.pt*, matheus.torquato@ifal.edu.br†, {denis, cerqueira}@ufpa.br‡, michele@dcc.ufmg.br§

***Abstract*—The Message Queuing Telemetry Transport (MQTT) protocol is a cornerstone of IoT communications. It relies on service brokers to enable reliable data delivery between devices and clients. In modern deployments, MQTT brokers are frequently hosted in virtualized environments to support scalability, flexibility, and resource efficiency. However, virtualization enlarges the attack surface, posing challenges to service reliability and security. This paper investigates the use of Virtual Machine (VM) migration as a Moving Target Defense (MTD) to enhance security in MQTT-based IoT services. While VM migration is an established technique in network and service management for workload balancing and fault tolerance, its impact, when used as a proactive security mechanism in MQTT deployments, remains unexplored. This work shows a comprehensive performance evaluation of VM migration under both normal and active attack scenarios. The results demonstrate that the security benefits of VM migration come with minimal performance degradation, characterized by a modest effect size (Cohen D measure$<$0.5), thus ensuring service continuity and operational stability. However, it comes with a cost of increased performance oscillation (i.e., higher incidences of peaks in the response time). This paper also introduces an interactive, web-based tool that enables pre-deployment MTD simulation. This work offers insights into integrating security-aware VM migration within IoT service management.**

***Index Terms*—VM migration, MQTT, Moving Target Defense, Internet of Things, Performance Evaluation**

## I. Introduction

The Message Queuing Telemetry Transport (MQTT) protocol [1] is one of the most widely adopted communication standards in the Internet of Things (IoT) ecosystem. It operates on a publish-subscribe model, where a central broker is responsible for managing and forwarding messages between clients, such as sensors and data consumers. In order to support scalability and reliability, MQTT brokers are often deployed in cloud-based, virtualized environments. However, this reliance on virtualization also enlarges the attack surface, exposing brokers to potential threats stemming from vulnerabilities in the underlying infrastructure.

A previous attempt to tackle this problem [2] followed the *execution throttling* (ET) approach. It limits the attacker's available resources to undermine attack progress. Prior research also has explored the use of Virtual Machine (VM) migration as a defensive mechanism to enable Moving Target Defense (MTD) strategies in virtualized environments [3]. However, there is a lack of analysis on how VM migration affects service performance both under normal operating conditions and during full-scale attack scenarios, where no safe or unaffected resources are available for migration.

This context motivates the investigation to understand the central research question (RQ): ***What is the performance impact in adopting VM migration as a defensive mechanism for MQTT brokers?*** Hence, this paper presents a comprehensive experimental study with three scenarios: (i) measuring migration-induced downtime, (ii) assessing performance impacts in an attack-free environment, and (iii) analyzing system behavior under a full-scale attack, where all physical resources are targeted. The attack model is based on Memory Denial of Service (memDoS) [4] (Section II-B provides more details).

The contributions of this paper are the following:

- The first and more straightforward is the assessment of managing VM migration-based MTD to MQTT performance, with a distinguishing aspect: under a *full-scale memDoS* attacks – a serious threat in constrained environments where attackers leverage co-location strategies to intensify the attack.
- The second contribution is the evaluation of VM migration downtime using a purpose-built, open-source monitoring tool[1], with results indicating consistent system downtime levels regardless of attacker presence.
- As a third contribution, we provide a readily available web-based tool[2] for custom evaluations.

[1]https://github.com/matheustor4/MigrationDowntimeAnalysis
[2]https://github.com/matheustor4/webAppPerfEvaluation2

This research was conducted within the framework of the SATERA project, being funded by the SESAR 3 Joint Undertaking (JU) under grant agreement No 101164313. The JU is supported by the European Union's Horizon Europe research and innovation programme, as well as by the SESAR 3 JU members other than the Union. This research was also supported by the INCT of Intelligent Communications Networks and the Internet of Things (ICoNIoT) funded by CNPq (proc. 405940/2022-0), Coordenacão de Aperfeicoamento de Pessoal de Nível Superior – Brasil (CAPES) Finance Code 88887.954253/2024-00, and by CISUC/LASI, through national funds by FCT - Fundação para a Ciência e a Tecnologia, I.P., in the framework of the Project UIDB/00326/2025 and UIDP/00326/2025.

978-3-903176-75-1 ©2025 IFIP

This paper proceeds as follows. Section II details the experimental setup. Section III documents the experimental results and analysis, highlighting the key takeaway points. Section IV provides an overview of the related works. Section V discusses some threats to validity and approaches for their mitigation. Section VI shows the web app details to simulate the attack-defense scenario. Section VII closes the paper, presenting conclusions and possible future research directions.

## II. Experimental Setup

This section presents the details of the testbed used for experimentation and has three subsections. Section II-A presents the architecture components and their configuration. Section II-B presents the details of the cybersecurity attack used in the experiments. Section II-C explains the adopted experimental approach for each studied scenario.

### A. System architecture

The experimental setup architecture (see Figure 1) has five components, namely: 1) Stresser - External machine generating the MQTT workload for the Victim VM; 2) Host A - Primary VM physical host; 3) Host B - Secondary VM physical host (bear in mind that VMs can migrate back and forth between Hosts A and B); 4) Victim VM - VM running the MQTT broker service; and 5) Attacker VM (1,2) - Attacker VMs conducting *memDoS* against Hosts A and B. Table I presents the host configurations.

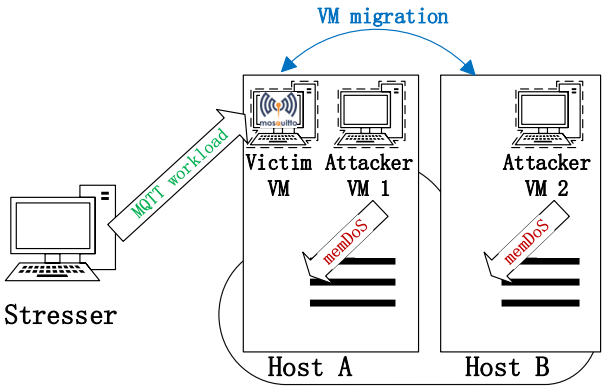



Fig. 1: System architecture - Environment used for the experiments

TABLE I: Host configuration summary

| Host | Hardware | OS/Config |
|---|---|---|
| Stresser | Intel Xeon E5-2620 2.0GHz 16 GB RAM | Linux Fedora 6.10.8-100 with MQTT benchmark tool [5]. |
| Host A | Intel Xeon E5-2620 2.0GHz 16 GB RAM | Ubuntu Server 20.04.6, kernel 5.4.0-195, KVM hypervisor 4.2.1 |
| Host B | Intel Core i7-9700 3.0GHz 16 GB of RAM | Ubuntu Server 20.04.6, kernel 5.4.0-195, KVM hypervisor 4.2.1 |

Additionally, the Attacker and Victim VMs are hosted on a Kernel Virtual Machine (KVM) hypervisor, with equal setups - single core vCPU and 3 GB of RAM with Ubuntu Server 20.04.2, kernel 5.4.0-190. The Attacker VMs run the *Memory Denial of Service* (*memDoS*) attacks (source available at [4]), and the Victim VM runs the Mosquitto MQTT Broker [6] version 1.6.9, with protocol version 3.1.1. Connectivity is provided by a TP-link TL-SG105 GbE Switch. We implement a python client-server script for VM migration downtime monitoring. The script consists of an User Datagram Protocol (UDP) client sending requests for the VM under migration. Then, we collect the *timeout* messages for proper downtime measurement.

### B. Attack model

The Attacker VM is running the *memDoS* attack [4], which overloads the internal VM memory. Due to isolation issues, the attack affects the VMs inside the same physical host (i.e., co-resident VMs) – this way, the attacker aims to produce a denial of service effect on VMs that are co-located with the Attacker VMs. In this case, we consider that, while under attack, both Attacker VMs are running the attack simultaneously.

The previous works applied VM migration to defend against *memDoS* [7], [8]. However, the VM migration was assumed to always arrive at an attacker-free host destination. In this paper, the approach is different since it is assumed that no safe places for migration exist while the system is under attack (i.e., *full-scale* attack). Since the VM migration MTD-based has already been evaluated as a defensive mechanism, the idea is to shift the focus to its potential performance impact under a worst-case scenario, shedding light on the pros and cons of adopting this approach for MQTT virtualized systems.

### C. Experiment design

The evaluation follows a set of experiments with four different scenarios. **Scenario #1) *No mig + No attack*** (NMNA) - This first scenario is the *baseline* scenario, that establishes what are the expected levels of MQTT broker performance without interference. **Scenario #2) *Mig + no attack*** (YMNA) - In this second scenario, the Victim VM is continuously migrating back-and-forth between Host A and Host B. Both Attacker VMs are running but idle. **Scenario #3) *No mig + attack*** (NMYA) - Victim VM stays at Host A without migration and under Attacker VM *memDoS*. **Scenario #4) *Mig + Attack*** (YMYA) - Victim VM migration between Host A and Host B. In this last scenario, both Attacker VMs are running the *memDoS* attack. At least 30 measurements were performed for each scenario.

The metric of interest in the experimentation is the mean per-message Round-Trip Time latency (i.e., time to send and receive a reply from an MQTT broker), which will be referred to as "response time" in the following sections for simplicity. The previous capacity planning [2] suggested that the proper (i.e., workload our system can respond with stable performance) MQTT workload is 100 clients publishing messages of 1500 bytes each. The comprehensive set of scenarios presented above provides a complete picture of the impact of VM migration in an MQTT attack-defense setup.

## III. Results and Analysis

This section presents and discusses the results for VM migration downtime (Section III-A) and the performance evaluation of the MQTT broker (Section III-B).

### A. VM migration downtime

The analysis of VM migration downtime is a recurrent subject in the literature [9], [10], with the majority of the research in the field being focused on minimizing system downtime during migration or evaluating it under different circumstances. Unlike these, the experiment hereby discussed aims to evaluate VM migration downtime while the technique is applied as MTD against a host-based attack. For measurement purposes, a custom *VM migration downtime monitor*[1] was implemented in Python. It consists of a lightweight User Datagram Protocol (UDP)-based client-server application in which the client requests the time from the server in an infinite loop, capturing timeout errors to detect possible interruptions in server activity.

As mentioned, the scope of the attack from the ATTACKER VMs is limited to the internal state of HOST A and HOST B. The network is not a target of the considered attack. Indeed, the traffic generated between the STRESSER and the VICTIM VM is due to the MQTT benchmark and *VM migration downtime monitor*. Two rounds of 30 VM migration operations were executed for each scenario: under attack (`Attack`) and without attack (`Idle`). Figure 2 presents a box plot comparing both scenarios, with Table II presenting the numerical results.

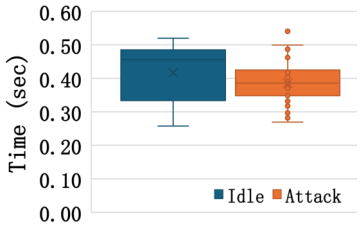



Fig. 2: VM migration downtime analysis boxplot. The blue box represents the results from the idle state (i.e., without attack) and the red box are the results from the experiment under attack.

TABLE II: Results of VM migration downtime experiment

| Experiment | Mean (sec) | Standard deviation (sec) |
|---|---|---|
| `Idle` | 0.417191 | 0.077689 |
| `Attack` | 0.380208 | 0.064294 |

The results suggest that **there is no statistically significant difference in the VM migration downtime due to the *memDoS* attack**, with the Mann-Whitney U test (i.e., one of the samples does not follow normality) failing to reject the null hypothesis (p-value of 0.1268), meaning there is no statistically significant difference between the samples. Thus, although VM migration affects system availability, the results suggest that the observed downtime does not seem to vary regardless of whether the system is under *memDoS* attack.

### B. MQTT performance evaluation

The MQTT performance evaluation results are into three groups: (1) Without Attack (i.e., Scenarios #1 and #2), (2) With Attack (Scenarios #3 and #4), and (3) Delay violation analysis. There is a brief discussion at the end of this section.

#### 1) Without attack - VM migration impact in absence of attack

The tests undertaken for this scenario aim at assessing the performance impact due to VM migration in our considered MQTT environment, thus establishing a comparison baseline for **applying VM migration in an attacker-free environment**. It is important to note that VM migration can also be triggered by other factors, such as preventive maintenance or server consolidation (i.e., packing VMs onto fewer physical machines to reduce power consumption). Time-based VM migration policies have also been proposed as a defense mechanism against zero-day attacks [11]. Therefore, gaining a deeper understanding of the impact of VM migration in an attacker-free MQTT environment is essential for informed deployment and operational planning.

Fig. 3 presents the result comparison for Scenario #1 (No mig + No attack) and Scenario #2 (Mig + No attack). The continuous line corresponds to the mean message time, the dotted line to the minimum message time, and the dashed line to the maximum message time. The X-axis corresponds to the collected measurements over time.

As expected, there is a disturbance in MQTT performance while the system is under continuous VM migration. The plots on Fig. 3 show that the mean and minimum values for both experiments (i.e., *No Mig + No attack* and *Mig + No attack*) are similar. However, the maximum value in *No Mig + No attack* experiment is below 40 ms, while the same value in *Mig + No attack* experiments peaks at 1555 ms. It suggests that, although the system experiences heavy oscillations during the continuous VM migration operations, it manages to roughly preserve a mean message time close to the baseline (i.e., *No mig + No attack*) conditions. Therefore, the maximum value peaks in *Mig + No attack* results seem to be outliers, as the mean values remain at levels similar to the baseline.

Fig. 4 presents a box plot comparison of the mean message time for both experiments to illustrate the disturbance effects better. It also contains a table with the mean and standard deviation of the two samples and the Cohen D effect size[3]. The obtained value was 0.32, suggesting that the VM migration produces only a *modest effect*[4] in the expected normal (i.e., baseline) results.

#### 2) Under attack - VM migration impact in presence of attack

While previous studies [3] have already demonstrated VM migration-based MTD effectiveness, the scenario assumes that a large-scale resource exhaustion attack (specifically, a *memDoS*) aiming at overloading the system is taking place. The tests undertaken for this scenario aim at evaluating the **VM migration-induced performance impact** in the MQTT environment **under a *memDoS* attack**. The goal lies in assessing if the system endures the combined attack and VM migration loads without experiencing a complete crash or failure. It also evaluates the resulting performance degradation in a VM-hosted MQTT system. This is relevant as VM migration —

[3]This is a measure of the relationship between two variables [12].

[4]The usual limits for the Cohen D effect size are: *small* ($d = 0.2$), *modest* (d = 0.5), *large* ($d = 0.8$), *very large* ($d = 1.20$), and *huge* ($d = 2.0$)

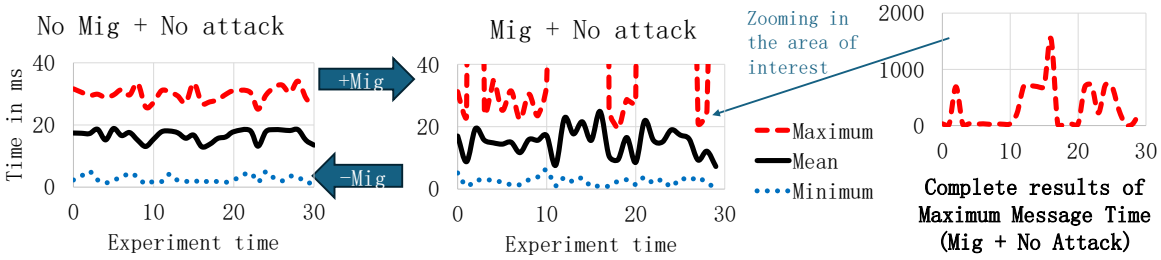
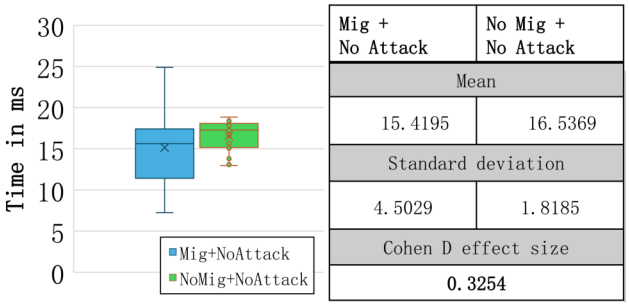



Fig. 3: Experiment results - Without attack Scenarios. The leftmost plot shows the results from the experiment without attack and migration. The middle plot presents a zoomed section of the rightmost plot with the data of interest for comparison. The rightmost plot presents the complete results of the maximum message time.




| Mig + No Attack | No Mig + No Attack |
|---|---|
| Mean | |
| 15.4195 | 16.5369 |
| Standard deviation | |
| 4.5029 | 1.8185 |
| Cohen D effect size | |
| 0.3254 | |

Fig. 4: Experiment results - Without attack Scenarios. The box plot presents the comparison between the results of both experiments. The table summarizes the desired statistics.

even in live mode [13] — that introduces a non-negligible resource overhead [14], which can aggravate the system strain during an ongoing attack. Fig. 5 shows the results.

The obtained results reveal unstable performance in both experiments. As with the baseline tests, comparing the mean values of experiments with and without VM migration does not capture the whole picture – however, the migration scenario produces significantly higher outliers (1614 ms with migration vs. 127 ms without migration). Fig. 6 presents a box plot of the mean message time for both experiments. Fig. 6 also presents the mean, standard deviation, and Cohen D effect size.

Overall, the results are similar to those of the previous experiment, indicating that **VM migration imposes a modest overhead on a system already under attack**. This finding complements prior research [8] – now incorporating scenarios of VM migration-based MTD in the context of an ongoing *full-scale memDoS* attack. Indeed, Mann-Whitney U Test[5] results in a $p - value$ of 0.189, indicating that the results there are no statistical differences between the two samples.

*3) Delay Violation - Impact in a hypothetical application*

The results presented provide a general aspect of the impact of VM migration in the mean response time. They highlight only a modest effect in the expected values. However, bringing additional context to enrich the proposed analysis is essential. For example, there are MQTT applications that have strict deadlines. Such cases highlight the maximum response time to verify possible acceptable delay violations.

[5]Used because one of the samples is not normally distributed

For this illustrative analysis scenario, let us consider an MQTT healthcare application. For this hypothetical scenario, suppose that the maximum allowed delay is 100 ms (arbitrarily defined, albeit based on [15]). From the result distribution, the maximum response time measurements made in each scenario are above the predefined acceptable delay. Fig. 7 presents a histogram for all four scenarios. The red dashed line represents the 100 ms threshold.

VM migration imposes heavy performance oscillations that lead the maximum values to surpass the predefined threshold. The results show that the scenarios with VM migration (prefix `YM`) have significant delay violations. We also computed the proportion of violations in each scenario for a better decision-making process. For that purpose, it *bootstraped*[6] the data to ensure fair comparison (see Table III).

TABLE III: Proportion of delay violations in each scenario

| Scenario | Estimated Proportion | | |
|---|---|---|---|
| | [95% CI - | Mean | 95% CI +] |
| **NMNA** | 0.0% | **0.4%** | 0.7% |
| **NMYA** | 16.0% | **28.0%** | 40.0% |
| **YMNA** | 29.2% | **41.7%** | 54.2% |
| **YMYA** | 67.7% | **80.6%** | 93.5% |

The presented results highlight that VM migration induces a higher incidence of response time peaks when compared to migration-less scenarios. Indeed, the migration itself (YMNA) brings heavier oscillation when compared with the attack itself (NMYA). Although VM migration has a protective effect, as demonstrated in the previous case study and prior work [8], its use in time-sensitive domains requires special caution.

## C. Takeaways

The analysis of the results presented in the previous section yields several key takeaways, which are next outlined.

*VM migration downtime does not present substantial variation due to memDoS ongoing attack:* We noticed that when the VM is under an indirect attack (*memDoS* targets the underlying host to affect the co-resident VMs), the VM migration downtime remains at the same levels of an attacker-free environment.

[6]Resampling method to assure fairness in the comparison by drawing random samples from the data. It uses a bootstrap sample size of 5000.

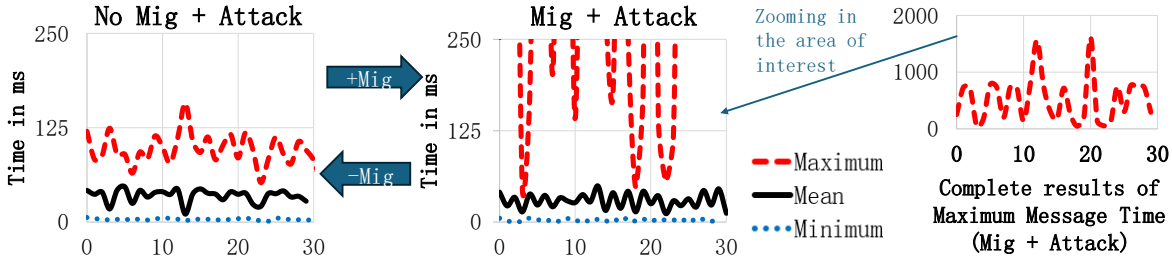
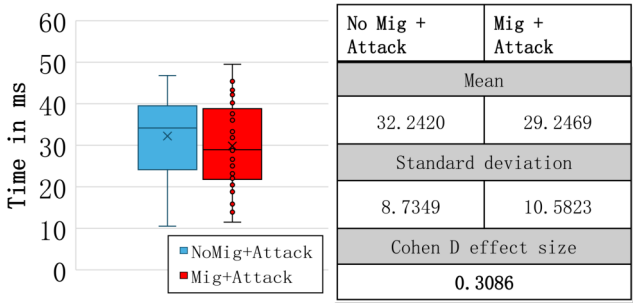



Fig. 5: Experiment results - With attack Scenarios. The leftmost plot shows the results from the experiment with attack and without migration. The middle plot presents a zoomed section of the rightmost plot with the data of interest for comparison. The rightmost plot presents the complete results of the maximum message time for the experiment with migration.




| No Mig + Attack | Mig + Attack |
|---|---|
| Mean | |
| 32.2420 | 29.2469 |
| Standard deviation | |
| 8.7349 | 10.5823 |
| Cohen D effect size | |
| 0.3086 | |

Fig. 6: Experiment results - With attack Scenarios. The box plot presents the comparison between the results of both experiments. The table summarizes the desired statistics.

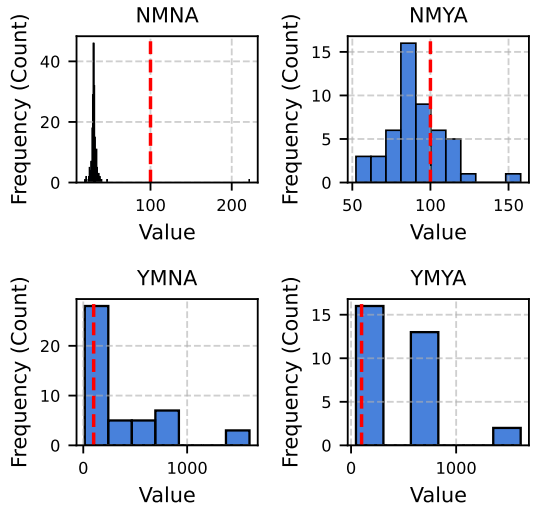



Fig. 7: Maximum response time result histograms for each scenario.

*VM migration produces a modest effect in the mean response time:* This result has further supported the adoption of VM migration for multiple purposes (e.g., system management, preventive maintenance, or server packing) in safe (i.e., attacker-free) environments. The results from the environment under attack show that the VM migration effect is also modest. These results show that VM migration-based MTD can preserve expected mean performance levels.

*VM migration as MTD in an MQTT time-sensitive scenario should be considered with extra caution:* Although the mean response time stays acceptable, the maximum values tend to achieve higher peaks compared to a migration-free approach. These results suggest that VM migration-induced peaks may lead the system to accumulate Service Level Agreement (SLA) violations. The general recommendation from our results is to carefully understand the time limitations of the particular deployments before enforcing continuous VM migration. However, it is important to highlight that our evaluation considers the worst-case scenario, with no safe host for migrations.

## IV. Related Works

Previous works [7], [8] have validated VM migration as a defensive technique. However, they neglect the possibility of a *full-scale* attack. The attacker may leverage co-location strategies or even replicate their VMs to increase attack power. Therefore, understanding how VM migration-based defense may affect the system is of utmost importance. Thus, the current paper offers a distinct perspective when compared to these previous works: 1) it examines the impact of VM migration specifically within an MQTT environment, and 2) it focuses on the performance implications of the technique rather than its effectiveness as an MTD strategy.

A relevant set of related works applies SDN techniques in the context of MTD for IoT. Zhou et al. work [16] merges MTD with cyber deception techniques to thwart cyberattacks. Swati et al. [17] present an SDN-powered MTD that leverages an intelligent traffic classifier to filter network packets, proposing a platform that also enables dynamic admission rules and resource remapping to ensure system availability. There are also techniques merging SDN and Game Theory to enhance MTD protection for IoT environments [18]. While our research used a different MTD strategy (i.e., VM migration), there is room for combining the proposed technique with SDN-based ones to improve overall MTD protection.

## V. Threats to Validity and Limitations

Below is a list of threats to the validity and limitations of this research, as well as suggestions for possible mitigation strategies. Due to space limitations, it is a non-exhaustive list.

*1) Limited testbed scale:* the considered testbed is minuscule when compared to large and more representative IoT deployments. Indeed, the collected VM migration measurements may (and likely will) vary in other environments,

thus calling for caution regarding their use for orchestration policy definition purposes. Nevertheless, the focus was to dismiss the idea that VM migration-based MTD is a universal strategy, calling attention to its limitations in certain situations. Additionally, the techniques and tools are publicly available, making it possible to adapt them accordingly to each scenario.

*2) The proposed technique is not specifically tailored for IoT.:* The threats for IoT are continuously evolving, the proposal of comprehensive MTD approaches is an appropriate strategy. Current IoT deployments tend to rely on cloud computing to integrate their numerous devices. Therefore, as cloud computing is a vulnerable point for IoT, this paper investigates the VM migration-based MTD under an IoT scenario.

*3) The representativeness of the selected threat model.:* This research is complementary for those dealing with other threats. Possibly, the provided web app may help the investigation of different policies against a broader scope of attacks. VM migration-based MTD is intended to be combined with other defensive methods to improve overall system security.

## VI. Web-Based Performance Evaluation Tool

We developed software to support the analysis of different scheduling approaches for attack and VM migration, where the transition between states of migration and attack can be user-driven, allowing him to exercise different VM migration scheduling scenarios against host-based attacks. Unlike our previous approach [7], the proposed tool offers the possibility of state transition based on the user decision (not only on a predefined model). Besides that, we adopted a web-based tool to enable the visualization of performance behavior during the experiment run. The tool provides a set of insightful outputs to support the decision-making process.

In summary, the tool acts as an environment simulator, based on pre-acquired data about the system behavior, to assess the expected performance when applying the user preferred policy, updating a plot dynamically based on the input data. The sections below show the details of the tool implementation. The tool source code was made available[7], as well as a demonstration[8].

### A. System state-machine

The system transits between four states based on the experiment scenarios (See Fig. 8). The intuition behind the tool development is to allow the user to analyze policies, where the system state changes upon their designed policy. The users can define and evaluate custom approaches before actual deployment.

In the states with activated migration, migration follows a continuous back-and-forth behavior between the available hosts (as presented in Section 1). Hence, while the system is in such states (prefix `YM` states), the expected performance behavior presents oscillations due to VM migration overhead.

[7]https://github.com/matheustor4/webAppPerfEvaluation2

[8]https://web-app-perf-evaluation2.vercel.app/

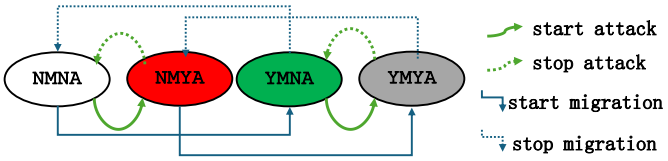



Fig. 8: System state-machine

*Quick note on data acquisition:* Data is from the laboratory specific hardware testbed configuration as input for the tool, thus being hard to generalize for other hardware configurations. The main recommendation is reproducing the performance evaluation experiment to gather appropriate data to feed the tool[9].

### B. Web Application User interface

The user interface (see Fig. 9) is organized into six areas.

1) **history** - stores the history of user-triggered state changes, including transition triggering time.
2) **state machine** - highlights the current system state of the simulation.
3) **plot area** - dynamically updated plot of the MQTT response time during the simulation experiment.
4) **delay panel** - input of acceptable delay and number of violations during the simulation experiment.
5) **state transition panel** - controls state transitions.
6) **cost panel** - input for: 1) expected operational cost per unit of time in each state; 2) cost per detected acceptable delay violation.
7) **simulation panel** - simulation control (the stop button ends the simulation and downloads the output).

### C. Tool outputs

The tool uses the performance evaluation data as input for plot generation. It has three outputs: `history`, `simulation_output` and `simulation_report`. The `history` file has the content of the **history** panel from the tool interface. The `simulation_output` file contains the hypothetical performance evaluation results from the policy that the user exercised in the tool (i.e., raw numerical data from the **plot area**). The `simulation_output` file also presents the acceptable delay and accumulated violations. The `simulation_report` is a one-page PDF file that offers a comprehensive picture of the exercised policy. It contains the generated plot and the history of state changes. Besides that, it adds a perspective of how much time the system spent in each state (i.e., sojourn time). Finally, it also presents an economic sustainability summary with the expected costs of the exercised scenario based on the input values of **cost panel**. Future development involves to add the *comparison feature*, enabling the user to compare different policies, and the *metrics calculator*, allowing for metric computation during the simulation runs.

[9]More guidance on replicating the experiment is on the repository page[2]

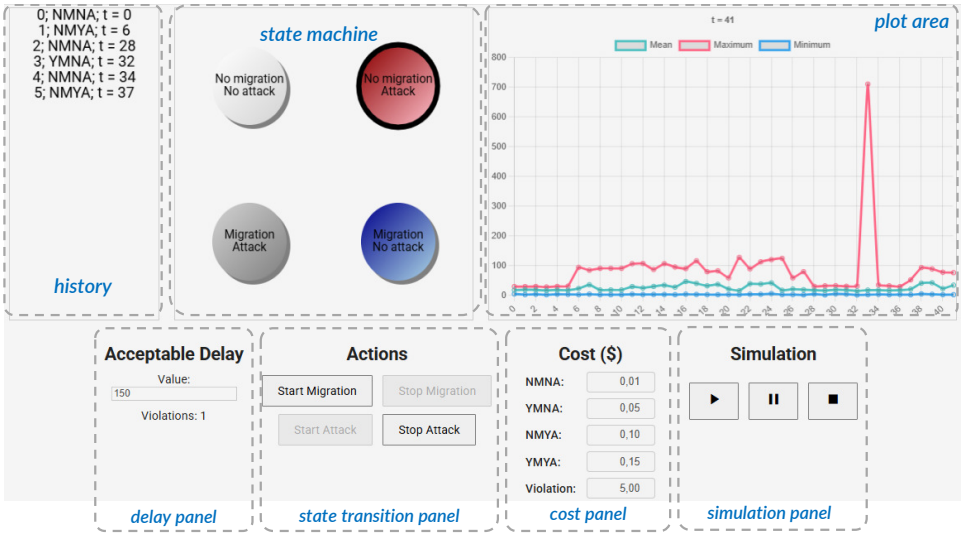



Fig. 9: Web-based tool interface. Annotations in blue. Better visualization at https://web-app-perf-evaluation2.vercel.app/

*A comment on tool flexibility for other scenarios:* As long as the user has performance data for the desired scenarios, it is possible to adapt the tool to act as a switching state game to obtain a hypothetical perspective of policy deployment, for example, in a redundancy allocation problem scenario. The user should collect data from the redundancy allocation alternatives and change the tool input files, thus allowing the analysis of the deployment with dynamic changes in the allocation approach using the state transition and the simulation panels.

## VII. Conclusion and Future Works

This paper presented a comprehensive performance evaluation of VM migration-based MTD against Memory DoS attacks. The study assessed the impact of VM migration on the performance of an MQTT broker instance, both under normal operating conditions and during an active attack. Experimental results showed that VM migration downtime remains comparable in both scenarios, with attack-free experiments indicating that the MQTT broker's performance is largely unaffected. However, for time-sensitive applications, VM migration significantly increases the number of deadline violations. These findings suggest that adopting VM migration as a defense mechanism requires careful consideration in attack scenarios. In general, the strategy is recommended only for systems with strict timing constraints when it is possible to guarantee a safe and attack-free target host. Future work will focus on integrating SDN-based MTD techniques into VM migration-enabled environments, investigating how their combination can strengthen overall MTD protection strategies.

## References

[1] G. C. Hillar, *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017.

[2] M. Torquato, B. Jesus, F. A. Silva, and E. Cerqueira, "Empirical observation of execution throttling as MQTT broker defense against memory denial of service attacks," in *Proc. of the 13th Latin-American Symposium on Dependable and Secure Computing*, 2024, pp. 184–187.

[3] T. Zhang, F. Kong, D. Deng, X. Tang, X. Wu, C. Xu, L. Zhu, J. Liu, B. Ai, Z. Han *et al.*, "Moving target defense meets artificial intelligence-driven network: A comprehensive survey," *IEEE Internet of Things Journal*, 2025.

[4] T. Zhang, Y. Zhang, and R. B. Lee, "DoS attacks on your memory in cloud," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 253–265.

[5] Krylovsk, "Krylovsk/mqtt-benchmark: MQTT broker benchmarking tool." [Online]. Available: https://github.com/krylovsk/mqtt-benchmark

[6] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017.

[7] M. Torquato, P. Maciel, and M. Vieira, "Evaluation of time-based virtual machine migration as moving target defense against host-based attacks," *Journal of Systems and Software*, vol. 219, p. 112222, 2025.

[8] M. Torquato and M. Vieira, "VM migration scheduling as moving target defense against memory DoS attacks: An empirical study," in *2021 IEEE Symposium on Computers and Communications (ISCC)*, 2021, pp. 1–6.

[9] F. Salfner, P. Tröger, and A. Polze, "Downtime analysis of virtual machine live migration," in *4th Int. Conf. on Dependability (DEPEND). IARIA*, 2011, pp. 100–105.

[10] N. Mukhopadhyay and B. P. Tewari, "Cost and energy aware migration through dependency analysis of vm components in virtual cloud infrastructure," *Computing*, vol. 107, no. 1, pp. 1–44, 2025.

[11] J.-H. Cho *et al.*, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 709–745, 2020.

[12] J. Cohen, *Statistical power analysis for the behavioral sciences*. Routledge, 2013.

[13] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. of the 2nd Symp. on Networked Systems Design & Implementation (NSDI'05) - Volume 2*, 2005, pp. 273–286.

[14] W. Voorsluys *et al.*, "Cost of virtual machine live migration in clouds: A performance evaluation," in *1st Int. Conference on Cloud Computing*. Springer, 2009, pp. 254–265.

[15] P. Akshatha and S. D. Kumar, "Delay estimation of healthcare applications based on MQTT protocol: a node-RED implementation," in *IEEE Int. Conf. on Electronics, Computing and Comm. Tech.*, 2022, pp. 1–6.

[16] Y. Zhou, G. Cheng, and S. Yu, "An SDN-enabled proactive defense framework for DDoS mitigation in IoT networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5366–5380, 2021.

[17] Swati, S. Roy, J. Singh, and J. Mathew, "Securing IIoT systems against DDoS attacks with adaptive moving target defense strategies," *Scientific Reports*, vol. 15, no. 1, p. 9558, 2025.

[18] M. Priyadarsini and P. Bera, "SDN deployed secure application design framework for IoT using game theory," *AI-Based Advanced Optimization Techniques for Edge Computing*, pp. 317–340, 2025.