VM Migration Scheduling as Moving Target Defense against Memory DoS Attacks: An Empirical Study

Matheus Torquato*†, Marco Vieira*
*University of Coimbra, CISUC, DEI, Coimbra, Portugal
†Federal Institute of Alagoas, Campus Arapiraca, Arapiraca, Brazil
matheus.torquato@ifal.edu.br, mdmelo@dei.uc.pt*†, mvieira@dei.uc.pt*

***Abstract*—Memory Denial of Service (DoS) attacks are easy-to-launch, hard to detect, and significantly impact their targets. In memory DoS, the attacker targets the memory of his Virtual Machine (VM) and, due to hardware isolation issues, the attack affects the co-resident VMs. Theoretically, we can deploy VM migration as Moving Target Defense (MTD) against memory DoS. However, the current literature lacks empirical evidence supporting this hypothesis. Moreover, there is a need to evaluate how the VM migration timing impacts the potential MTD protection. This practical experience report presents an experiment on VM migration-based MTD against memory DoS. We evaluate the impact of memory DoS attacks in the context of two applications running in co-hosted VMs: machine learning and OLTP. The results highlight that the memory DoS attacks lead to more than 70% reduction in the applications' performance. Nevertheless, timely VM migrations can significantly mitigate the attack effects in both considered applications.**

***Index Terms*—Memory DoS, Moving Target Defense, VM migration, Dynamic platform technique, Denial of Service**

I. Introduction

Moving Target Defense (MTD) consists of dynamically changing the available attack surface to thwart or defend from attacks [1], [2]. In virtualized environments, Virtual Machine (VM) migration appears among the most used MTD strategies [3], being preferred among other MTD techniques for several reasons, including i) VM migration is usually a native feature of virtualized environments; ii) it does not require expertise to deploy; iii) it is already a usual task of the virtualized environment management.

VM migration consists of moving the VMs in the available physical machines (PM) [4]. In the MTD context, VM migration is frequently applied to prevent malicious VMs from affecting the co-resident VMs or the underlying PM [5], [6] (i.e., prevent host-based attack success). For example, it is possible to use VM migration-based MTD to move benign clients away from a compromised PM [7].

Specifically, on the threats with potential defense through VM migration, we highlight the memory Denial of Service (memory DoS). Leveraging on issues in the hardware memory isolation [8], the attacker can run an attack against the memory of his own VM, trying to affect the co-resident VMs availability by overloading memory.

One of the critical factors to deploy an effective MTD against memory DoS is timing (i.e., when to apply MTD) [9]. In fact, in the context of VM migration against memory DoS, the MTD timing is still an open problem. Previous research tried to tackle this issue through stochastic modeling [10], [11]. However, these works neglect empirical investigation of VM migration-based MTD against memory DoS. Zhang et al. [12] refers to VM migration as a potential defense for memory DoS, but the authors use an alternative mitigation method based on *execution throttling*.

This practical experience report aims to fill this research gap through an experiment on VM migration-based MTD against memory DoS. We intend to investigate the impact of different scheduling of VM migration in the potential MTD protection. The following research questions guide this research:

- RQ_1*:* What is the impact of memory DoS in different applications running on co-resident Virtual Machines?
- RQ_2*:* Is Virtual Machine migration effective as Moving Target Defense against memory DoS?
- RQ_3*:* Does the Virtual Machine migration scheduling policy play a significant role in the mitigation of memory DoS effects?

The experimental approach is as follows. First, we set up an environment with two VMs, ATTACKER VM and VICTIM VM, running inside the same physical host. While the ATTACKER VM runs memory DoS attacks, the VICTIM VM runs benign applications, namely, an online transaction processing (OLTP) application benchmark and a machine learning (ML) application. Then, as MTD, we migrate the ATTACKER VM at different scheduling times, observing five-minute intervals. Note that these five-minute intervals are selected arbitrarily to

This work has been partially supported by Portuguese Foundation for Science and Technology (FCT), through the PhD grant SFRH/BD/146181/2019, within the scope of the project CISUC - UID/CEC/00326/2020. This work is also funded by the European Social Fund, through the Regional Operational Program Centro 2020.

This work also received support from AIDA: (Adaptive, Intelligent and Distributed Assurance Platform) project, funded by Operational Program for Competitiveness and Internationalization (COMPETE 2020) and FCT (under CMU Portugal Program) through grant POCI-01-0247-FEDER-045907. And, from project TalkConnect funded by COMPETE 2020 trough grant POCI-01-0247-FEDER-039676

978-1-6654-2744-9/21/$31.00 ©2021 European Union

fit our experiment design. Here, we focus on understanding the VM migration-based MTD effectiveness against memory DoS instead of defining generic methods for selecting VM migration intervals. Besides that, the search for the critical threshold for VM migration is out of the scope of this paper.

Our results show that the memory DoS attacks performed do not interfere in the ML application accuracy, and effects are only noticeable in the ML time to fit metric[1]. In the ML application scenario, the VM migration is enough to clear the memory DoS effect). Regarding the OLTP application, we notice that delayed migration leads to cumulative service degradation. However, in all studied scenarios, the OLTP application stayed alive during the attack (i.e., the attack is not enough to crash the application). We present linear regression curves to help characterize the VM migration-based MTD protection in the context of this OLTP application.

This paper tries to fill a research gap by providing empirical evidence of the VM migration-based MTD effectiveness against memory DoS. We can highlight the following:

- Easy-to-reproduce experimental approach. We try to describe our methodology in detail to help researchers and system managers to reproduce the experiment in their environments. All the tools and source code are publicly available.
- We investigate the impact of memory DoS attacks in two different applications, namely OLTP and machine learning. From the investigation, we present a comprehensive set of results providing evidence of the effectiveness of VM migration-based MTD against memory DoS attacks.
- We consider the effects of an attack that is easy-to-launch, challenging to detect, and causes substantial performance degradation. Therefore, our study may help system managers to handle this relevant security threat.

The remainder of this paper is organized as follows. Section III presents the experiments. Section II presents the related work. Section IV presents the results. Section V briefly discusses the specific memory DoS severity. Section VI presents threats to validity and limitations of our work. Section VII concludes the paper.

II. Related Works

Our previous works [10], [11] provide VM migration as MTD evaluation based on modeling. These papers neglect experimental validation for the models. This practical experience report extends these works by providing the needed empirical background of VM migration as MTD.

The inspiring work of Zhang et al. [12] provided memory DoS background (including the attack source code). Li et al. [8] provided insights on the memory DoS attack detection. Although both papers mentioned VM migration as a potential defense for memory DoS, the authors followed different approaches from ours. Zhang et. al. [12] dealt with the problem using *execution throttling*. Li et al [8] focused on the memory DoS detection instead of MTD proposal.

Wang et al. [13] proposed a comprehensive framework for defending against co-resident threats. Their framework features a score calculation and attack-aware VM reallocation. Likewise, Liang et al. [14] defensive approach consists of a grouping-based VM placement strategy. In both papers, the authors validated their approaches using CloudSim. Unlike their works, we decided to deploy VM migration in a real testbed. Besides that, instead of proposing a new framework, our goal is to observe how the *off-the-shelf* VM migration MTD scheduling may protect the considered applications.

III. Experimental approach

Our main goal is to assess the impact of a memory DoS attack on applications running in co-resident VMs. Besides that, we aim to study whether different VM migration scheduling policies effectively mitigate possible effects of memory DoS.

Figure 1 presents the experimental testbed, which includes two physical machines: Source Node (Intel Xeon E5-2620 2.00GHz + 16GB of RAM with Error Correction Code enabled) - main host for the Victim VM and Attacker VM; Target Node (Intel Core i7-9700 3.00GHz + 16GB of RAM) - host for the Attacker VM migration. Both the Attacker VM and the Victim VM are Kernel Virtual Machine (KVM)[2] VMs with a homogeneous configuration: single-core processor + 3 GB of RAM. The Source Node and the Target Node run Ubuntu Server 20.04.2 with kernel 5.4.0-72 and KVM 4.2.1.

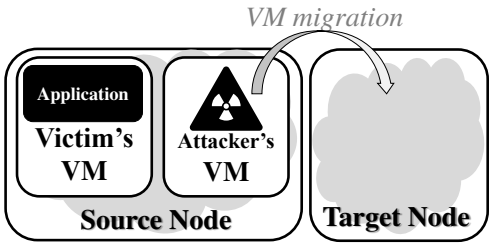

Fig. 1. Testbed architecture

We consider two different applications running in the Victim VM. The first consists of an ML application for face recognition based on an example of scikit learn python library[3]. We provide an example script for the ML application automation in [15], which produces an output file with the accuracy and time to fit metrics. The second is an OLTP application based on the TPC-C[4] benchmark [16]. Specifically, we use the CockroachDB tool [17] for TPC-C benchmark automation.

Regarding the specific attack running in the Attacker VM, we follow the approach presented by Zhang et al. [12]. In

[1]The fitness function is the main task of our ML application. Indeed, the time spent on other tasks is negligible. Thus, the time to fit is roughly the ML processing time.

[2]https://www.linux-kvm.org/

[3]https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html

[4]http://www.tpc.org/tpcc/

practice, we use an infinite loop of unaligned atomic accesses to the main memory of the VM. This attack load generates LOCK signals in the memory, whose accumulation may result in memory unavailability to handle benign processes. Henceforth in this paper, we refer to this attack as *unalignAttk*.

We performed sets of 30-minutes experiments, during which the ATTACKER VM performs *unalignAttk* attacks and the VICTIM VM runs the OLTP or the ML application. For comparison purposes, we present results for three scenarios: i) *golden run* - experiments without attacks and MTD; ii) *MTD* - experiment adopting VM-migration as MTD against *unalignAttk*; and iii) *Only attack* - considering the *unalignAttk* impact while the MTD is off.

IV. CASE STUDIES

This section presents our two case studies. As mentioned earlier, in these case studies, we consider two different applications running inside the VICTIM VM, namely a machine learning application and an OLTP application benchmark (TPC-C benchmark).

A. Machine Learning application

We divided the experiment with the ML application into two steps. First, the *attack severity experiment*, focusing only on investigating the impact of *unalignAttk* on the application (i.e., system without MTD). Second, the *MTD experiment*, where we apply VM migration scheduling as MTD. The former aims to provide an answer to *RQ1* (impact of memory DoS in different applications running on co-resident Virtual Machines), and the latter provides an answer for *RQ2* (Virtual Machine migration effective as Moving Target Defense against memory DoS) and for *RQ3* (role played by the Virtual machine migration scheduling policy in the mitigation of memory DoS effects).

The results (see Figures 2 and 3) include the *golden run* (system without attack) and the *unalignAttk* (system under attack). The X-axis corresponds to the experiment timeline.

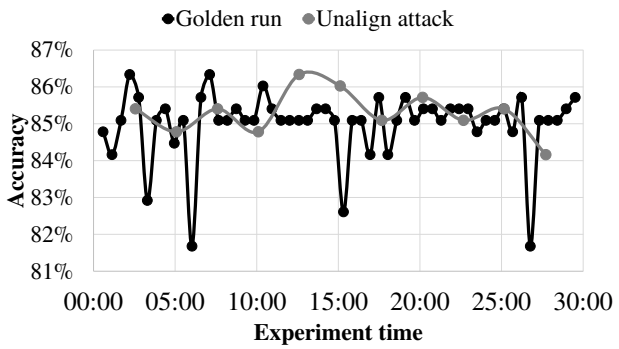

Fig. 2. ML - Accuracy results - attack severity experiment

Figure 2 shows that accuracy is close to 85% for all the ML observations. In both curves (*golden run* and *unalignAttk*), we notice the expected accuracy oscillations over time. These results suggest that considering the scope of our experiments, the *unalignAttk* does not interfere with the accuracy of the ML application. Therefore, as long as the system is up, even in the presence of an *unalignAttk* attack, the ML application results accuracy stays roughly at the same levels of the *golden run*.

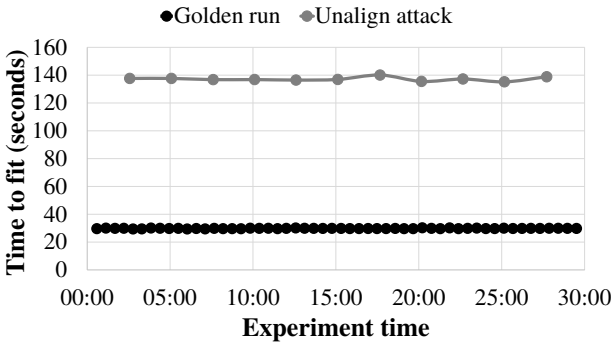

Fig. 3. ML - Time to fit (seconds) results - attack severity experiment

Time to fit results in Figure 3 show how long the ML application takes to process the face recognition. These results highlight a substantial difference between the *golden run* and the *unalignAttk*. Indeed, the ML application processing under *unalignAttk* lasts about four times more than the *golden run*. These results suggest that the *unalignAttk* impairs the ML application performance substantially, reducing the overall number of ML application runs in a given period. Although the *unalignAttk* does not cause a catastrophic failure of ML application, its impact may reflect in the application availability. Depending on the Service Level Agreements (SLA) and some threshold levels, the ML application may be considered unavailable when it takes so long for processing.

In summary, we noticed that the major impact is indeed in the performance and not in the ML accuracy. The *unalignAttk* causes a 460% increase in the time to fit when compared to the *golden run*. The number of ML runs in our 30-minute experiment is of 54 in the *golden run* and 11 in the *unalignAttk*, meaning a 80% reduction. Table I presents a summary of ML application *attack severity experiment*.

The second step of this experiment is the *MTD experiment*, in which we deploy three different schedules of VM migration, namely at the 5th, 10th, and 15th minute of the experiment time. In the same way as the *attack severity experiment*, we noticed that among the regular oscillations, the accuracy results for all the MTD scenarios also approach 85%. Nevertheless, the time to fit results presents more interesting behavior as shown in Figure 4.

These results suggest two conclusions. First, the VM migration MTD is effective to clear *unalignAttk* effects. Second, delayed or premature VM migrations immediately recover the ML application to the *golden run* levels. However, the longer the attack continues, the worse is the performance impact. Note that the ML application is a standalone application without a timeout parameter. VM migration timing is crucial for system availability in more complex client-server scenarios,

TABLE I
ATTACK SEVERITY RESULTS SUMMARY - MACHINE LEARNING APPLICATION

Experiment	Number of runs	avg. accuracy	std. dev. accuracy	avg. time to fit	std. dev. time to fit
golden run	54	85.00%	0.0091	29.7702	0.18283
unalignAttk	11	85.29%	0.0058	137.18	1.31916

TABLE II
MTD RESULTS SUMMARY - MACHINE LEARNING APPLICATION

Experiment	Number of runs	avg. accuracy	std. dev. accuracy	avg. time to fit	std. dev. time to fit
OnlyAttack	11	85.29%	0.0058	137.18	1.3192
MigAt5thMin	46	85.15%	0.0072	34.92	22.6595
MigAt10thMin	39	84.93%	0.0109	41.79	32.9165
MigAt15thMin	33	85.22%	0.0053	49.17	40.5078

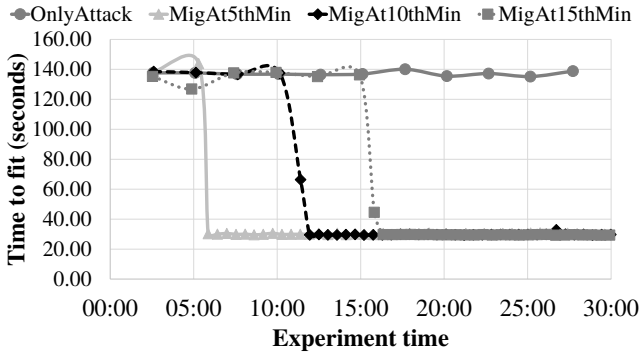

Fig. 4. ML - Time to fit (seconds) results - MTD experiment

as delayed VM migrations may allow the client to accumulate server timeout, leading the client to give up the connection.

Table II presents a summary of the ML application *MTD experiment*. The high standard deviation variance in the time to fit MTD scenarios is due to the abrupt change in the time to fit after VM migration.

B. TPC-C benchmark

Unlike the ML application experiment, where we collect the metrics from every ML run, a 30-minute experiment run provides only a single result of the TPC-C benchmark. Therefore, we need to run a set of 30-minutes TPC-C evaluations. The following results are obtained from 30 runs of *golden run* and 30 runs of *OnlyAttack* (i.e., system without MTD). We run 15 experiments for the MTD results (three for each migration schedule).

Here, we focus in two metrics: **efc(%)** - how close the results are to the theoretical maximum TPC-C performance, and **avg (ms)** - average time for transaction processing in milliseconds. To these, we added two metrics: **tpmC** - TPC-C specific metric to measure the *business throughput* (i.e., number of orders processed per minute), and **ops** - total number of transactions processed.

In the TPC-C benchmark results, we merged all the scenarios into the same plot. This approach is helpful to notice the degradation due to delayed migrations. Therefore, the plots have *golden run* results at the origin, meaning the VM migration at the 0th minute (i.e., system without attack), and *OnlyAttack* results at 30th minute (i.e., system without MTD as each experiment lasts 30 minutes). We perform experiments with VM migration at the 5th, 10th, 15th, 20th, and 25th minute of experiment time.

Figures 5 and 6 presents the results for *avg (ms)* and *efc(%)*, respectively. The plots include the error bars for each scenarios and a linear regression curve. In both cases, the linear regression curves $R - squared$[5] is above 0.99.

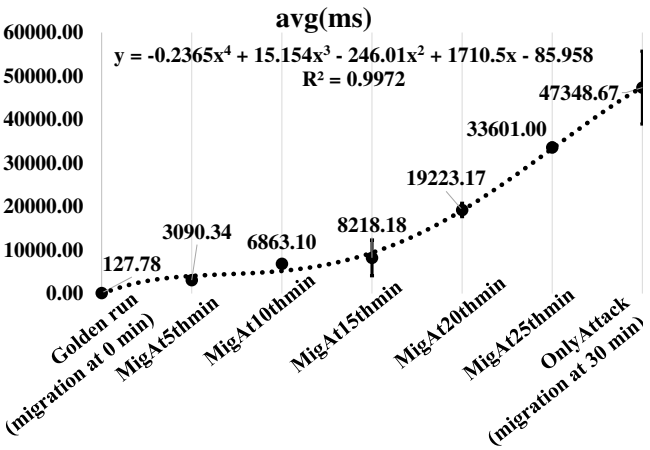

Fig. 5. TPCc - avg (ms)

We notice the expected behavior of *avg (ms)* increasing and *efc* decreasing when we have delayed MTD actions. Specifically about our first research question related to memory DoS impact (i.e., comparison between *golden run* and *Only Attack*), the decrease in the *efc* is of 68.39%, while the increase in the *avg (ms)* approach 370%. These results reveal the service degradation rate due to memory DoS attack effects accumulation.

The linear regression curves are particularly useful to estimate *avg (ms)* and *efc* with other intervals for VM migration. For illustration, we obtain that, to preserve *efc* above 75%, VM migration should occur before 7.5 minutes, and to keep

[5] $R - squared$ (R^2) is a measure that corresponds to the proportion of the variance explained by regression curve.

TABLE III
TPC-C BENCHMARK EXPERIMENT RESULTS

Experiment	avg. tpmc	std. dev. tpmc	efc(%)	std. dev. efc(%)	avg (ms)	std. dev. avg (ms)	ops	std. dev. ops
golden run	123.5	0.4814	96.00	0.0038	127.78	9.4855	8554	31.5369
MigAt5thMin	105.2	1.7688	81.81	1.3809	3090.34	597.9909	7281	102.9208
MigAt10thMin	91.5	1.0497	71.40	0.8155	6863.10	157.8724	6358	75.8650
MigAt15thMin	65.2	6.3168	50.68	4.8976	8218.18	4123.8594	4542	445.1576
MigAt20thMin	58.9	4.4500	45.77	3.4567	19223.17	1545.3677	4060	305.2223
MigAt25thMin	46.7	1.1518	36.30	0.9092	33601.00	983.7309	3273	75.5484
OnlyAttack	35.5	6.5807	27.61	5.1120	47348.67	8384.8285	2461	446.3500

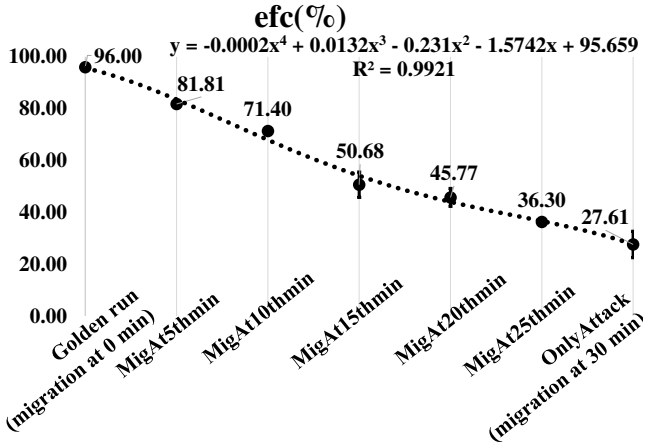

Fig. 6. TPCc - efc(%)

the *avg (ms)* below 15000, VM migration should occur before the 18th minute.

Table III summarizes the TPC-C benchmark results. It includes the results for the metrics and their standard deviation. Specifically about tpmC, one of the metrics of interest in the TPC-C benchmark, the reduction is 71.25% (comparison between *golden run* and *OnlyAttack*).

V. SEVERITY OF *UnalignAttk* ATTACKS

In the previous section, we highlighted the *unalignAttk* impact on the applications. However, it is crucial to investigate whether the effect is only due to a manageable resource overhead. The question (Q) is *Isn't the impact observed already expected due to unalignAttk resource overhead?*.

To answer Q, we propose an observation of *unalignAttk* resources overhead and its comparison with benign workloads. As we are dealing with memory, we set up a workload based on the Linux memtest[6]. We also added the workloads considered in the case studies, namely, TPC-C and Machine learning (ML) applications. In practice, we observed the SOURCE NODE resource consumption while hosting one VM running the workload and one VM in idle state. The VM's configuration is the same used in the previous case studies. The considered workloads are: *golden run* (i.e., two VMs in idle state), *unalign attack*, *memtest*, *TPC-C* and *machine learning application (ML)*. The observation in all scenarios lasts 30 minutes. The results from the CPU and memory usage are presented in Figures 7 and 8, respectively.

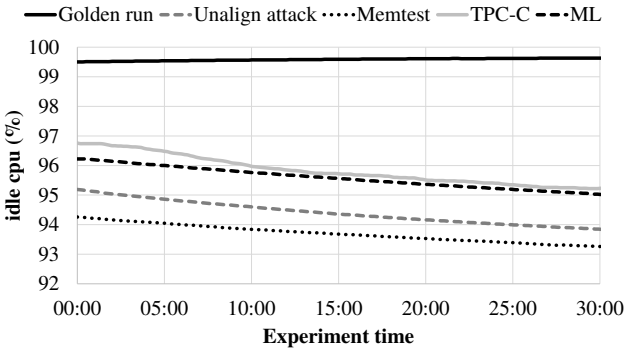

Fig. 7. Percentage of idle CPU

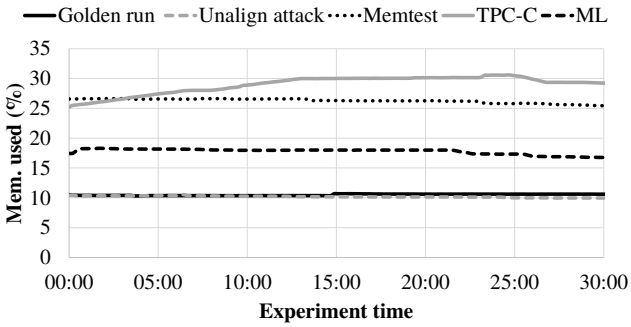

Fig. 8. Memory usage (%)

We notice that, *memtest* requires more SOURCE NODE resources than the *unalignAttk*. Presumably, considering only the resource consumption, the possible impact of *memtest* in the VICTIM VM should be higher than the *unalignAttk* impact. To verify this, we run four one-hour experiments combining the *memtest* and *unalignAttk* running in the ATTACKER VM with the ML application and the TPC-C benchmark running in the VICTIM VM. Table V and IV show the results.

The results highlight that the *unalignAttk* produces a higher impact on the applications when compared to the *memtest*. Actually, the *memtest* results are close to the *golden run* results. While the *memtest* causes a reduction (when compared to the *golden run*) of 22.2% in the TPC-C benchmark *efc* metric, the *unalignAttk* produces a 79.1% reduction. About the ML application, *memtest* reduces the number of runs in 7.5%,

[6]https://linux.die.net/man/8/memtester

TABLE IV
COMPARISON BETWEEN *unalignAttk* AND *memtest* - TPC-C METRICS

Attack	tpmC	efc	avg (ms)	ops(total)
golden run	124.4	96.7%	126.4	17202
memtest	95.9	74.5%	7385.3	13263
unalign	22.6	17.6%	82225.5	3131

TABLE V
COMPARISON BETWEEN *unalignAttk* AND *memtest* - ML METRICS

Attack	Number of runs	Avg. accuracy	Avg. time to fit(s)
golden run	40	85.05%	29.87
memtest	37	85.10%	34.02
unalign	19	84.88%	125.26

while the *unalignAttk* reduction is of 52.5%. The results of this experiment confirms the findings of [12] which highlights memory DoS attack severity.

VI. THREATS TO VALIDITY AND LIMITATIONS

We identified two main threats to the validity of our results, and they lie in our experiment design: 1) the observed results are obtained from a small architecture; 2) limited observation time in all the experiments. In the best scenario, we should run more extended experiments in bigger datacenters. We are aware of these limitations. However, below we provide some explanations for mitigating these threats.

Threat 1) We manage to dedicate a small but powerful setup for our experimentation. Note that the PMs have 16GB of RAM with 6 and 8-core processors. To scale a representative scenario, our VMs have only 3 GB of RAM each, with a single-core processor. Therefore, there are plenty of idle PM resources while running the VMs simultaneously, taking less than 40% of the available resources.

Threat 2) Note that each experimentation does not involve only the 30 minutes of the workload. We need to clean the system for each run of the experiment, meaning generate new VM images, complete PM OS reboot, export filesystem for VM migration, and create new VMs. Besides that, all the experiment runs are sequential, which means that it was impossible to paralleling the experiment runs (as we have only a single testbed).

VII. CONCLUSION

This paper presented a practical experience report of VM migration scheduling as MTD against memory DoS attacks. We evaluated the memory DoS attack severity and the MTD effectiveness in different scenarios. Namely, we considered memory DoS attacks against i) a machine learning application and ii) the TPC-C benchmark. Our results show that the memory DoS attack causes a significant impact on the applications. Besides that, results suggest that the VM migration-based MTD effectively reduces the effect of memory DoS attacks in both applications.

This work fills a research gap of lack of empirical evidence of VM migration-based MTD effectiveness against memory DoS. We are aware that the results are limited to our system architecture. However, the adopted tools and source code are publicly available. Thus, it is possible to reproduce the proposed methodology in other scenarios. Hopefully, system managers and researchers may find our approach useful to support MTD experimentation and MTD policy design.

In the future, we aim to reproduce the experiments in a bigger datacenter comprising other representative workloads as client-server applications. Besides that, we intend to run more extended experiments to notice possible error accumulation after sequenced VM migration.

REFERENCES

[1] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving target defense: creating asymmetric uncertainty for cyber threats*. Springer Science & Business Media, 2011, vol. 54.

[2] 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.

[3] M. Torquato and M. Vieira, "Moving target defense in cloud computing: A systematic mapping study," *Computers & Security*, vol. 92, p. 101742, 2020.

[4] P. G. J. Leelipushpam and J. Sharmila, "Live vm migration techniques in cloud environment—a survey," in *2013 IEEE Conference on Information & Communication Technologies*. IEEE, 2013, pp. 408–413.

[5] H. Wang, F. Li, and S. Chen, "Towards cost-effective moving target defense against ddos and covert channel attacks," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, 2016, pp. 15–25.

[6] T. Penner and M. Guirguis, "Combating the bandits in the cloud: A moving target defense approach," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2017, pp. 411–420.

[7] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell, "Catch me if you can: A cloud-enabled ddos defense," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 264–275.

[8] Z. Li, T. Sen, H. Shen, and M. C. Chuah, "Impact of memory dos attacks on cloud applications and real-time detection schemes," in *49th International Conference on Parallel Processing-ICPP*, 2020, pp. 1–11.

[9] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A survey of moving target defenses for network security," *IEEE Communications Surveys & Tutorials*, 2020.

[10] M. Torquato, P. Maciel, and M. Vieira, "Security and availability modeling of vm migration as moving target defense," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2020, pp. 50–59.

[11] ——, "Analysis of vm migration scheduling as moving target defense against insider attacks," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 194–202.

[12] 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.

[13] X. Wang, L. Wang, F. Miao, and J. Yang, "Svmdf: A secure virtual machine deployment framework to mitigate co-resident threat in cloud," in *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1–7.

[14] X. Liang, X. Gui, A. Jian, and D. Ren, "Mitigating cloud co-resident attacks via grouping-based virtual machine placement strategy," in *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2017, pp. 1–8.

[15] M. Torquato and M. Vieira. (2021). [Online]. Available: https://github.com/matheustor4/scriptML

[16] S. T. Leutenegger and D. Dias, "A modeling study of the tpc-c benchmark," *ACM Sigmod Record*, vol. 22, no. 2, pp. 22–31, 1993.

[17] CockroachDB. (2021). [Online]. Available: https://www.cockroachlabs.com/docs/v20.1/performance-benchmarking-with-tpc-c-10-warehouses