

# HERO vs. Zombie: Identifying Zombie Guests in a Virtual Machine Environment

Yael Elinav<sup>1</sup><sup>a</sup>, Alex Moshinky<sup>1</sup><sup>b</sup>, Lior Siag<sup>1</sup><sup>c</sup> and Nezer Jacob Zaidenberg<sup>1,2</sup><sup>d</sup>

<sup>1</sup>Faculty of Computer Science, College of Management, 2 Eli Weisel st., Rishon LeZion, Israel

<sup>2</sup>Faculty of IT, University of Jyväskylä, Jyväskylä, Finland

**Keywords:** Virtualization Sprawling, Inactive Virtual Machines, Virtual Machines, Remote Management, System Administration, Green Computing.

**Abstract:** Virtual servers are important in many data-centers. Multiple guest virtual machines are consolidated on several hosts on-site or on the cloud, and serve the organization's computational needs. However, virtual machines not cleared from the system, known as zombie machines, waste resources and pose a security risk. We present a novel tool to optimize resource use by tracking down zombie machines: HERO (Host Environment Resource Optimization). HERO leverages multiple testing approaches and machine learning to assist system administrators in locating "zombie" machines.


## 1 INTRODUCTION


Server virtualization is a crucial part of meeting modern organizations' computing needs. Server virtualization allows for higher reliability, improved flexibility, and reduced costs compared with traditional physical servers (Steinder et al., 2007). Virtualization also improves server security by facilitating patch management (Ray and Schultz, 2009). In fact, virtualization and related technologies such as SGX and TrustZone (Zaidenberg, 2018) are key factors in Industry 4.0 security. However, the ease of adding new virtual machines introduces new problems. Virtualization sprawling (Suchithra and Rajkumar, 2012) is a phenomenon that occurs when the virtual environment grows to a scale too large to be managed effectively by the system administrator. Virtualization sprawling can cause security risks (Luo et al., 2011) and administration overhead (Carroll et al., 2011), and causes valuable system resources to be wasted, e.g., from storage and RAM allocation to increased electricity use and CO2 emissions (Belanger and Casemore, ). "Zombie" VMs (virtual machines), or comatose VMs, (Kooimey and Taylor, 2017) are a great contributor to this phenomenon. A zombie VM is a


virtual machine that is no longer needed yet is still active and consuming resources. A server can turn into a zombie for many reasons. For instance, a system administrator may forget to eliminate an experimental VM. In another example, whenever a new system replaces an outdated VM, the outdated VM can be discarded. However, the system team often keeps the older system intact to ensure fast rollback if a problem occurs (Colman-Meixner et al., 2016) on the new system. The system team may forget to delete these older systems long after the new system is considered as stable as the old one. Another cause for unattended VMs is swapping employees and workplaces. Sometimes, VMs that belonged to former employees are left unattended as current system staff forgot about their existence or purpose and do not want to risk compromising operations. The number of zombies varies from company to company and can comprise up to 30% of VMs in a given cloud environment (Mazumdar and Pranzo, 2017). Removing them can improve resource allocation at no additional cost. In addition, these servers incur considerable costs through increased power consumption and security risks as discussed in (Sapp, 2014) and others sources


This paper suggests a solution to help address the growing problem of virtualization sprawling.

The primary purpose of our system is to discover zombie VMs using an accurate, efficient, and reliable method. The program that we introduce, HERO (Host

<sup>a</sup> <https://orcid.org/0000-0001-6722-3602>

<sup>b</sup> <https://orcid.org/0000-0002-0571-0066>

<sup>c</sup> <https://orcid.org/0000-0002-5517-5006>

<sup>d</sup> <https://orcid.org/0000-0003-3496-7925>

Environment Resource Optimization), tests the guest VMs hosted by a hypervisor. HERO performs multiple tests to determine with high probability which VMs are zombie VMs. In this paper, we describe those tests and the weight mechanism that adjusts the test result to the unique characteristics of the data center in which HERO runs.

## 2 HERO SYSTEM OVERVIEW

We built HERO using Python3.8, collecting and parsing the data using Python and shell scripts. We developed HERO for the KVM (kernel virtual machine) (Habib, 2008) hypervisor. KVM is an open source virtualization technology and HERO can support multiple KVM servers from one instance. All the VMs used to write the program use the Ubuntu operating system and run Prometheus (Padgham and Winikoff, 2002) to publish raw data via HTTP.

HERO needs to operate from a server with access to the VMs in the KVM environment and a user to run KVM “virsh” (Kovari and Dukan, 2012) commands. HERO can run on a VM inside the KVM hosts. The HERO program contains four key modules:

**Discovery.** This module is responsible for the discovery and extraction of information about the VMs. An additional program, Prometheus, is used to make the information available for further processing. This is according to Prometheus methodology (Padgham and Winikoff, 2005).

**Inspection.** This module contains all tests run on the VMs and the resulting data. HERO uses this data to determine what VMs might be zombies.

**Action.** This module contains the activities HERO performs on suspected zombies. These activities are listed online for the system administrator.

**Training.** This module reviews the data obtained from VMs and confirmed as zombies by the system administrator in order to adjust the weight of each test to reflect the characteristics of the zombies remaining in the cloud environment.

### 2.1 Discovery Module

We used Node exporter (Großmann and Schenk, 2018) and wrote custom scripts for those metrics. The scripts periodically read and parse the data published by the node-exporter to allow VM testing. The files containing the parsed data are continuously updated and limited by size. Therefore, the files usually contain data from approximately the last two weeks of operation to ensure relevance.

### 2.2 Inspection Module

HERO runs several tests on every VM. The tests return a number that represents the level of confidence that the machine is indeed a zombie VM. This grade will be later multiplied by a weight that represents how significant the test is (e.g., the “name test” is less valuable than the “CPU test”). The sum of the final grades is the score of the VM: how strongly HERO “believes” it is a zombie. Zombie VMs can act in different ways and operational VMs may have several zombie characteristics. Therefore, we chose to conduct a large number of tests and give them varying significance.

### 2.3 Tests

It is often quite easy to verify that a server is not a zombie. For example, port TCP-22 (ssh) traffic or changing levels of CPU usage would indicate that a VM is currently in use and operational. No network traffic or close to zero CPU usage for a long period of time would indicate that a VM is indeed a zombie.

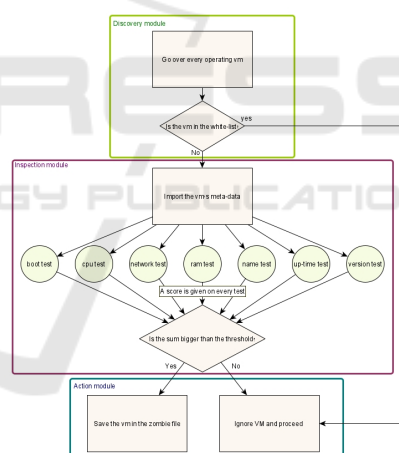


Figure 1: HERO architecture.

A server that fails to communicate with external sources for long periods or that reboots frequently and fails to demonstrate the stability expected from production servers are likely to be zombies. However, such servers may simply be temporarily off-duty (for example, during off-peak periods). HERO will attempt to confirm which machines are zombies and which are legitimate virtual machines merely temporarily idled.

1. CPU Test: HERO checks the CPU usage of the VM. A server that is idle most of the time is likely not operational. CPU usage varies when the VM only runs default services versus initiated

programs. The test checks the 15-minute average CPU load against a configured threshold and returns the percentage of those occurrences. (Bila et al., 2012) used similar tests to detect idle systems.

2. **RAM Test:** HERO checks how often the VM allocates RAM. A server that is idle most of the time is likely not operational and its RAM usage varies when the VM only runs default services versus initiated programs. This test works similarly to the CPU test. A similar method was used in (Bila et al., 2012).
3. **Network Test:** HERO checks how communicative the VM is. As VMs are likely to generate network traffic regardless how operational they are (e.g., ARP, broadcast, DHCP) (Georgiou et al., 2013), this test checks if the daily incoming and outgoing network traffic exceeds a certain threshold. The test score is how often the VM has not communicated with another device (transmit or receive). Some VMs may not even have a network card at all, and cannot be reached. Such VMs are highly likely to be zombies.
4. **Uptime Test:** A server having a very long uptime means no significant upgrades or periodic maintenance shutdowns were performed on the server (Kim et al., 2016). A larger-than-configured number of uptime days will result in a bad score (a zombie candidate).
5. **Frequent Booting Test:** Frequent rebooting is an abnormal behavior that will be quickly noticed and fixed in a non-zombie VM. This test will determine the number of times the VM has rebooted in the documented period (Galante and de Bona, 2012).
6. **Name Test:** Suspicious names (e.g., “test,” “check,” “abc,” “try,” “temp”) usually indicate that a VM is temporary. Production VMs most likely have a more meaningful name.
7. **Software Version Test:** HERO checks if the VM’s software version is included in the “approved versions” list. This can be done using agents or by examining the RAM of the inspected system using LibVMI (Payne, 2012), NSX (Pettit et al., 2018), or similar systems. A VM with an old version will indicate an unmaintained and rarely upgraded VM. This is a very important test because VMs with older operating system versions are unlikely to have recent security patches, meaning they can very easily be a security threat.

## 2.4 Action Module

HERO provides the system administrator with a list of suspected zombies in the environment. HERO also provides a detailed results file for specific virtual machines. Thus, HERO allows the system administrator to better understand how a given VM is operating and to perform analysis on the reasons it was suspected. After reviewing the suspected zombies, the system administrator can list the actual zombie virtual machines. These VMs can be used as a control group for retraining and improving the accuracy via the Training Module, and then later deactivated.

## 2.5 Training Module

Given that the profile of a zombie machine varies between data centers, we wanted to adapt the program to the needs of individual data centers. One data center may define a zombie machine as one that did not communicate for a long time, whereas a different data center may be interested to find machines that are not using the CPU allocated to them. To accommodate HERO to different needs, we used a rule-based training module with linear programming and a CNN-based learning module. The system administrator can enter a list of confirmed zombie virtual machines as a control group. HERO will run tests both on the suspected and real zombie virtual machines, and will increase or decrease the weights accordingly. For each test, HERO will check if the median test result of the control group is higher or lower than that of the suspected group for each test.

## 3 RESULTS

We tested HERO in a lab environment, hosting 8 zombie VMs and 25 working VMs. The workload was created by scripts that initiated random network communication and used CPU and RAM for long periods.

At the first executing of HERO, with the weights that we chose, the following results were obtained: True Positive Rate (TPR) = 0.363 and True Negative Rate (TNR) = 0.818, with balanced accuracy of 0.590. After training HERO once (using training code written by us) and raising the threshold, the following results were obtained: TPR = 0.454 and TNR = 0.863, with balanced accuracy of 0.658. The results show that HERO was able to learn and adjust the weights according to our test lab environment.

## 4 RELATED WORK AND SYSTEMS

Currently, there are some monitoring tools that inspect virtual environments and only a few that address the zombie VM problem. Here we describe a few of them and their merits, and why they do not completely solve the zombie problem. Some similar solution includes iCSI. ICSI(Kim et al., 2017) is a garbage collector based on resource utilization decision model. It identifies the purpose of each VM first, as differently purposed VMs have different utilization patterns, and thus different way to determine inactivity. It then searches for active VMs, as they have stronger affinity towards those patterns. It performs network affinity analysis for the VMs marked as inactive in order to reduce the number of VMs wrongly marked as inactive. Another utilization-based model (Fesl et al., 2019) adds client-VM network activity and screen changes into consideration. This approach says that if there are many changes to the screen, there is a user interacting with the system. Likewise, if there are many packets sent between the client and the VM, it is likely that the VM is currently in use. If both do not indicate activity, it refers to CPU, memory, and network usage to determine the level of activity. Grabo (Cohen and Bremler-Barr, ) creates a directed graph with edges representing dependency between resources. The user inputs a set of core resources which are used with non-cloud dependencies, and those are used as the roots of a Mark & Sweep process. This approach fails to consider stand-alone VMs, which can be active, just not connected to other resources. Pleco (Shen et al., 2016) is another example for the graph-based approach. It creates a dependency graph between applications and resources, assigns weight for each reference which represents how much does the connection indicates activity, and calculates confidence level for the VMs marked as inactive. (Kim et al., 2016) Attempts to identify inactive virtual machines by checking different parameters. They establish that most VMs have low resource utilization. They added login history check into their parameters testing. Established that different VMs have different purpose and thus different resource usage patterns and such each group needs to be evaluated differently. Their model was trained to find active VMs from the VMs after filtering the obvious inactive ones. They did that because active VMs show stronger features to determine there are active than inactive VMs to determine they are inactive. The group that is left is then the inactive ones. This can be cited as a direct competing approach to HERO. (wook Baek et al., 2014) is an implementation of similar system.

This one simply has a mechanism that deletes VMs nobody accessed for a determined amount of time. It does not take other parameters into consideration. This can be cited as another example of inactive VM detection and garbage collection, even if it's a very simple one.

SolarWinds Virtualization Manager (Kedia et al., 2013) provides high-quality and reliable virtual machine monitoring, performance management, capacity planning, and optimization. SolarWinds can control VM sprawl and resource inefficiencies by powering off idle VMs or deleting VMs no longer in use. This can help to reclaim resources that the host machine or other VMs can then use. It is able to find some zombie VMs, but this tool only works on VMWare and Hyper-V. It also requires an expensive license. RVTools (Mauro et al., 2017) is a Windows application that can display information about the virtual environment. RVTools is able to list information about VMs, CPU, memory, disks, partitions, networks, and much more. It is able to find some zombie VMs, but this tool only works on VMWare ESXs virtualization servers. Prometheus is an open-source system monitoring and alerting toolkit originally built at SoundCloud. Prometheus (Padgham and Winikoff, 2002) is a full monitoring and trending system that includes built-in and active scraping, storing, and more. It has knowledge about what the world should look like (which endpoints should exist, what time series patterns mean trouble, etc.) and actively tries to find faults. Though it has extensive VM monitoring, it does not have a specific zombie machine finder. WhatsUp Gold (Hernantes et al., 2015) is a monitoring performance and availability end-to-end tool. WhatsUp Gold displays hosts and guests, host/guest relationships, clusters, and the real-time status. It also monitors the performance and resource consumption of hosts and guests, including CPU, memory, disk, and more, but it does not have a specific zombie machine finder and only works in VMware and Hyper-V environments. Veeam One (Graziano et al., 2013) provides historical data to forecast resource usage and plans for capacity changes and is able to model usage trends and predict costs for storage, compute power, and backup. It can find possible infrastructure weaknesses or vulnerabilities, but it does not have specific zombie machines finder and only works in VMware and Hyper-V environments. SolarWinds as one of the strongest monitoring tools in the market and advertises its VM utilization as one of its most important features. As shown above, most of the monitoring tools available today work only with VMWare and Hyper-V, some do not look for zombie VMs, and most of them charge licensing fees. These factors em-

phasize the need for a project such as ours. Another way to detect zombies is by using an introspection solution, such as libvmi (Payne, 2012) or NSX (Pettit et al., 2018), or software inside the VM (Kiperberg et al., 2019). The memory can later be analyzed using tools such as rekall (Block and Dewald, 2017) or volatility (Graziano et al., 2013). While these systems may provide all the required components to detect zombies, they do not include a unique tool and some hacking may be required. In addition, the performance cost is quite significant. The system operated using Linux servers and workstations, but with the exception of new tests and the inspection module the system can operate equally as well using Windows, OS X, MVS, or any other enterprise operating system.

## 5 CONCLUSIONS

We presented HERO, a system designed to locate and identify zombie VMs running on the inspected host or cluster. We have used HERO on KVM environments, though the system can be ported easily to Hyper-V, ESXi, Acropolis, and Xen.

We designed and developed HERO, a system to locate and identify zombie VMs on the inspected host or cluster. By deploying HERO in a virtual server environment, we found that HERO could successfully locate VMs and improve zombie detection accuracy through the Training Module, thus validating our approach and use of machine learning.

Unfortunately, we could not access a live data center to experiment on and, therefore, had to estimate the common zombie characteristics based on our knowledge and experience with real-life production environments. Because HERO cannot definitively verify which VMs are zombies, system administrators must still confirm which VMs to deactivate. However, with further development, HERO could prove to be an invaluable tool for system administrators, empowering them to quickly and accurately track down zombie machines and, thus, optimize computing capacity and save costs.

## ACKNOWLEDGEMENTS

We thank the College of Management, Academic Studies, for a research grant that allowed us to develop and test the system described in this paper.

## REFERENCES

- Belanger, S. and Casemore, B. "exploring the impact of infrastructure virtualization on digital transformation strategies and carbon emissions" an idc white paper, sponsored by vmware.
- Bila, N., de Lara, E., Joshi, K., Lagar-Cavilla, H. A., Hiltunen, M., and Satyanarayanan, M. (2012). Jettison: Efficient idle desktop consolidation with partial vm migration. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 211–224.
- Block, F. and Dewald, A. (2017). Linux memory forensics: Dissecting the user space process heap. *Digital Investigation*, 22:S66–S75.
- Carroll, M., Kotzé, P., and Van der Merwe, A. (2011). Secure virtualization: benefits, risks and constraints.
- Cohen, N. and Bremler-Barr, A. Graph-based cloud resource cleanup.
- Colman-Meixner, C., Develder, C., Tornatore, M., and Mukherjee, B. (2016). A survey on resiliency techniques in cloud computing infrastructures and applications. *IEEE Communications Surveys & Tutorials*, 18(3):2244–2281.
- Fesl, J., Gokhale, V., and Feslová, M. (2019). Efficient virtual machine consolidation approach based on user inactivity detection. *CLOUD COMPUTING 2019*, page 115.
- Galante, G. and de Bona, L. C. E. (2012). A survey on cloud computing elasticity. In *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, pages 263–270. IEEE.
- Georgiou, S., Tsakalozos, K., and Delis, A. (2013). Exploiting network-topology awareness for vm placement in iaas clouds. In *2013 International Conference on Cloud and Green Computing*, pages 151–158. IEEE.
- Graziano, M., Lanzi, A., and Balzarotti, D. (2013). Hypervisor memory forensics. In *International Workshop on Recent Advances in Intrusion Detection*, pages 21–40. Springer.
- Großmann, M. and Schenk, C. (2018). A comparison of monitoring approaches for virtualized services at the network edge. In *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, pages 85–90. IEEE.
- Habib, I. (2008). Virtualization with kvm. *Linux Journal*, 2008(166):8.
- Hernantes, J., Gallardo, G., and Serrano, N. (2015). It infrastructure-monitoring tools. *IEEE Software*, 32(4):88–93.
- Kedia, P., Nagpal, R., and Singh, T. P. (2013). A survey on virtualization service providers, security issues, tools and future trends. *International Journal of Computer Applications*, 69(24).
- Kim, I. K., Zeng, S., Young, C., Hwang, J., and Humphrey, M. (2016). A supervised learning model for identifying inactive vms in private cloud data centers. In *Proceedings of the Industrial Track of the 17th International Middleware Conference*, pages 1–7.

- Kim, I. K., Zeng, S., Young, C., Hwang, J., and Humphrey, M. (2017). icsi: a cloud garbage vm collector for addressing inactive vms with machine learning. In *2017 IEEE International Conference on Cloud Engineering (IC2E)*, pages 17–28. IEEE.
- Kiperberg, M., Leon, R., Resh, A., Algawi, A., and Zaidenberg, N. (2019). Hypervisor-assisted atomic memory acquisition in modern systems. In *International Conference on Information Systems Security and Privacy*. SCITEPRESS Science And Technology Publications.
- Koomey, J. and Taylor, J. (2017). Zombie/comatose servers redux. Report by Koomey Analytics and Anthesis. Recuperado de <http://anthesisgroup.com/zombie-servers-redux>.
- Kovari, A. and Dukan, P. (2012). Kvm & openvz virtualization based iaas open source cloud virtualization platforms: Opennode, proxmox ve. In *2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics*, pages 335–339. IEEE.
- Luo, S., Lin, Z., Chen, X., Yang, Z., and Chen, J. (2011). Virtualization security for cloud computing service. In *2011 International Conference on Cloud and Service Computing*, pages 174–179. IEEE.
- Mauro, A., Valsecchi, P., and Novak, K. (2017). *Mastering VMware vSphere 6.5: Leverage the power of vSphere for effective virtualization, administration, management and monitoring of data centers*. Packt Publishing Ltd.
- Mazumdar, S. and Pranzo, M. (2017). Power efficient server consolidation for cloud data center. *Future Generation Computer Systems*, 70:4–16.
- Padgham, L. and Winikoff, M. (2002). Prometheus: A methodology for developing intelligent agents. In *International Workshop on Agent-Oriented Software Engineering*, pages 174–185. Springer.
- Padgham, L. and Winikoff, M. (2005). Prometheus: A practical agent-oriented methodology. In *Agent-oriented methodologies*, pages 107–135. IGI Global.
- Payne, B. D. (2012). Simplifying virtual machine introspection using libvmi. *Sandia report*, pages 43–44.
- Pettit, J., Pfaff, B., Stringer, J., Tu, C.-C., Blanco, B., and Tessmer, A. (2018). Bringing platform harmony to vmware nsx.
- Ray, E. and Schultz, E. (2009). Virtualization security. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, pages 1–5.
- Sapp, K. L. (2014). *Managing Virtual Infrastructure with Veeam® ONE™*. Packt Publishing Ltd.
- Shen, Z., Young, C. C., Zeng, S., Murthy, K., and Bai, K. (2016). Identifying resources for cloud garbage collection. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 248–252. IEEE.
- Steinder, M., Whalley, I., Carrera, D., Gaweda, I., and Chess, D. (2007). Server virtualization in autonomic management of heterogeneous workloads. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 139–148. IEEE.
- Suchithra, R. and Rajkumar, N. (2012). Efficient migration—a leading solution for server consolidation. *International Journal of Computer Applications*, 60(18).
- wook Baek, H., Srivastava, A., and Van der Merwe, J. (2014). Cloudvmi: Virtual machine introspection as a cloud service. In *2014 IEEE International Conference on Cloud Engineering*, pages 153–158. IEEE.
- Zaidenberg, N. J. (2018). Hardware rooted security in industry 4.0 systems. *Cyber Defence in Industry 4.0 Systems and Related Logistics and IT Infrastructures*, 51:135–151.