Defining a measure of adversarial strength for co-residency based attacks on container orchestration systems

Sushrut Shringarputale, Patrick McDaniel

***Abstract*—Co-resident cloud systems, despite their assurances, have always been vulnerable to direct and indirect attacks by malicious systems. Current cloud systems provide two options: either accept the status quo or pay more money for an isolated environment. The introduction of sophisticated cloud orchestration systems like Kubernetes have made it much easier to manage what application is running on what physical machine. In this paper, we demonstrate that despite cloud-provider security infrastructure, it is possible to prove co-residency or a victim container on state-of-the-art Kubernetes systems. However, each orchestrator generates enough noise in hardware access to affect co-residency detection and attacks. We define a measure for adversarial strength on such systems for a particular type of attack and compare the strength of adversaries against various orchestrators.**

I. Introduction

CLOUD computing has grown exponentially in the last decade, growing to $175 Billion in revenue in 2018 [1]. The adoption of cloud computing has also grown to a large swath of industries including healthcare, finserv, technology, insurance, and more [2]. From small startups to large multinational corporations, everyone seems to have some data on the cloud, a large portion of which can be highly sensitive. A major attack surface for customers of the cloud comes from having their code run on the same physical machine as a possible adversary (co-residency), leading to active privacy leaks. Despite the introduction of sophisticated management platforms such as Kubernetes, these threats still continue to exist. In light of Google's recent announcements CitationNeeded [3] for making clusters co-resident at the process isolation level rather than the virtual machine layer makes the work in this paper even more important. In this paper we will

- Demonstrate based on work by Bates et al. [4] how co-residency can still be an issue on modern cloud systems
- Define a measure of adversarial strength (platform resiliency) against co-residency detection and compare three major cloud orchestration platforms against such attacks
- Investigate the reason behind the disparities between such systems and propose solutions along those lines to improve resiliency against co-residency

II. Background

According to NIST [5], cloud computing is defined as

> "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Cloud service providers (CSP) such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform etc. provide certain services to their customers with a certain set of guarantees (service level agreements). While it is evident to customers that they are sharing the physical hardware with other customers, hereby referred to as tenants, the cloud platform also provides certain isolation guarantees for any code running on these systems. More specifically, Linux processes provide isolation guarantees that ensure that the memory in use is completely isolated from other processes. Access Control mechanisms in the operating system provide isolation at file and device levels. Similarly, hypervisors provide isolation to virtual machines running complete operating systems. More recently, containers are replacing virtual machines as the standard compute unit on the cloud. All of these isolation strategies have been proven to be vulnerable to various side-channel attacks [6] [7] [8].

A. Threat model

A co-resident attack considers an active, malicious adversary that has no affiliation to the cloud provider. The adversary appears as an innocuous tenant to the CSPs with no elevated privileges on the actual machine they are logged onto. They are free to launch as many instances on the cloud service as they want (barring some limits for specific CSP) and can choose the hardware configuration the instances will have. This hardware configuration is applied at the VM level, however, certain CSPs tend to restrict the type of VM instance that can be placed on a specific type of machine [3]. This means that the adversary has the capability to choose from a subset of physical machines in some cases. For example, if an adversary decides to use **m2.large** instances on AWS, they will most likely be placed on machines specified for **m2.large** instances [9]. The adversary also has access to the standard interface for launching instances, same as the victim.

The victims are legitimate cloud tenants performing some security sensitive operations using hardware that is possibly shared with the adversary. The victim has the same capabilities as the adversary when it comes to launching instances. Both members in this model trust the cloud provider and have little to no insight into the placement policies of the datacenter. Additionally, the adversary gains little to no information from metadata provided to the instance such as the IP address. While this information was useful in older attacks for cloud cartography [10], this is no longer viable on current cloud architectures.

Instead, the adversary aims to use one of the available side channels in order to gain unauthorized access to some information belonging to the victim. This information can range from knowledge of existence [4], [10] to active attacks such as stealing private keys from cryptography applications [7]. In this paper, we will use the watermarked network flow performance degradation side-channel to generate all our metrics.

B. Container orchestrators

A new type of isolation scheme at the process level is becoming the standard for running applications on cloud systems. This isolation makes use of Linux namespaces and other abstractions in order to virtualize process generation. Each process is made to believe that it is running on an independent machine with no interference, similar to a virtual machine. In order to mimic platform agnostic execution, all relevant libraries are bundled in to the process using a Layered File system (LayerFS) CitationNeeded [3]. All these facets are packaged into one ”container” and can be run quite easily. Since it is only a process and all libraries are bundled into the application ahead of time, the spinup time for containers tends to be very low CitationNeeded [3].

Modern cloud systems make use of containers running inside VM instances for fast launch, cross-platform compatibility, and security. More recently, large scale applications make use of containers to scale horizontally when demand rises. The fast spinup time makes for realtime and responsive scaling, as well as cost effective usage of compute resources. Additionally, more complicated applications can be broken into individual pieces called microservices which communicate with each other over the network. Each microservice can scale independently as the demand is generated without affecting its peers in any way.

Large scale applications are quickly adopting these strategies as it provides fast scaling, cost effectiveness, and lower downtimes due to rolling updates. In order to manage such architectures, there is a lot of work being put into applications that will manage the deployment of such microservices for very large use cases. These applications orchestrate the placement, scaling, and deployment of containers in a distributed setting providing capabilities to respond to large spikes in traffic and demand very quickly.

Sophisticated orchestration platforms to manage deployments using containers have opened the doors for more advanced container handling for cloud systems. Google’s Borg project led to the development of Kubernetes [11] which helps automate the process of assigning containers to physical machines. In addition, the platform performs various additional virtualization at the compute, device, and network levels to simplify application development. While Kubernetes is still vulnerable to co-residency based attacks, it may be possible to enhance the abstractions developed with that platform to minimize the risk of co-residency attacks.

III. Conclusion

The conclusion goes here.

References

[1] K. Costello and S. Hippold, "Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17.3 Percent in 2019," Sep. 2018. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-09-12-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2019

[2] L. Columbus, "Roundup Of Cloud Computing Forecasts And Market Estimates, 2018," Sep. 2018. [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2018/09/23/roundup-of-cloud-computing-forecasts-and-market-estimates-2018/

[3] CitationNeeded, "Citation needed." [Online]. Available: http://xkcd.com/

[4] A. Bates, B. Mood, J. Pletcher, H. Pruse, M. Valafar, and K. Butler, "Detecting Co-residency with Active Traffic Analysis Techniques," in *Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop*, ser. CCSW '12. New York, NY, USA: ACM, 2012, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/2381913.2381915

[5] P. M. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Tech. Rep., 2011. [Online]. Available: https://doi.org/10.6028/nist.sp.800-145

[6] Y. Yarom and K. Falkner, "FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack." in *USENIX Security Symposium*, vol. 1, 2014, pp. 22–25.

[7] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-Tenant Side-Channel Attacks in PaaS Clouds," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, 2014, pp. 990–1003. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2660267.2660356

[8] W. Zhang, X. Jia, C. Wang, S. Zhang, Q. Huang, M. Wang, and P. Liu, "A Comprehensive Study of Co-residence Threat in Multi-tenant Public PaaS Clouds," in *A Comprehensive Study of Co-residence Threat in Multi-tenant Public PaaS Clouds*, 2016, pp. 361–375. [Online]. Available: http://link.springer.com/10.1007/978-3-319-50011-9_28

[9] V. Varadarajan, Y. Zhang, T. Ristenpart, and M. Swift, "A Placement Vulnerability Study in Multi-Tenant Public Clouds."

[10] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds." ACM, Nov. 2009, pp. 199–212. [Online]. Available: http://dl.acm.org/citation.cfm?id=1653662.1653687

[11] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in *Proceedings of the European Conference on Computer Systems (EuroSys)*, Bordeaux, France, 2015.