Lind: Enhancing Container Security by Leveraging the Popular Paths

Yiwen Li

Abstract

Containers, such as Docker and LinuxKit, are widely used because of the perceived isolation and security they provide against the potentially malicious or buggy user programs running inside of them. However, this perception may be false. It is possible to trigger zero-day kernel bugs from inside of a container, which could lead to such security problems as privilege escalation. Containers remain vulnerable because they are allowed access to rarely executed paths in the host operating system kernel that often contains bugs. To solve this problem, we first developed a "popular paths" metric that revealed a strong correlation between the popularity of kernel code and the likelihood that it contains security bugs. Now we are applying our popular paths metric to real-world container systems, Docker and LinuxKit. We will further extend our system to automate the application of the popular paths metric to multiple kernel versions, and large-scale container applications.

Motivation

Operating system kernels form a critical trusted computing base in modern operating systems such as Linux. Bugs in the kernel may allow untrusted user code to escalate privileges, and potentially compromise the entire system, causing severe security problems to the system and any program running on it. Despite the importance of kernel security, commodity OS kernels such as Linux still have a large number of bugs. There were 176 CVE vulnerabilities found in the Linux kernel in 2018, and 67 vulnerabilities confirmed in the first six months of 2019 [5].

Lightweight virtualization systems (or containers) have recently become a popular way to deploy and run applications with their dependencies. In fact, one industry survey suggesting the technology could become a $2.7 billion market by 2020 [1]. This wide-spread adoption of containers, such as Docker [3] and LinuxKit [7] can be attributed to the perceived isolation and security they provide to the user programs running inside of them. However, this perception may be false, as it is possible to trigger zero-day kernel bugs from inside of a container. And since the kernel is shared by containers and the host system, exploitation of such bugs could lead to severe security problems, such as privilege escalation. For example, an exploit against CVE-2016-5195 (the "Dirty COW" vulnerability) was demonstrated that allowed attackers to escape from a Docker container and access files on the host system [2].

To prevent containers from triggering security vulnerabilities in the host kernel, we need to have an effective way of knowing where bugs are located first, and then use our knowledge of the bug locations to monitor and prevent user programs from reaching these risky code paths.

Goals

The ultimate goal of our work is to improve security of virtualization systems and containers by preventing them from triggering kernel vulnerabilities. To do so, we first establish a quantitative security metric to predict the locations of zero-day kernel bugs. We then study and demonstrate the feasibility of applying our metric to real-world Docker containers. Lastly, we will automate and generalize the application of our metric to multiple kernel versions, and large-scale container applications.

Roadmap

Step one: finished

The first step in our work is to find an effective metric in predicting where bugs are likely to be located. This can help us understand how to avoid triggering security vulnerabilities, and thus reduce the attack surface. Finding a good metric lays the foundation of our work.

In our previous research work, we have discovered a powerful correlation between the popularity of a kernel code path—how often it is encountered during the course of normal execution—and the likelihood that it contains a security flaw [6]. In a study of 40 Linux kernel bugs, we found that just one occurred in one of these "popular" paths. Furthermore, this correlation was statistically significant $(p < .01)$. Our popular paths metric is therefore a powerful predictor of where security-critical bugs are likely to occur. The study and development of our "popular paths" metric has published in our "Lock-in-Pop" paper at the USENIX ATC '17 [6].

Step two: on-going

In our current on-going research, we are studying the feasibility of using the popular paths metric to secure the LinuxKit virtual machine (VM). This entails first, finding a way to identify and capture the popular paths for the LinuxKit VM, then, demonstrating that existing containers could operate their normal / default workload on these paths, and finally, developing a method to warn users away from code found in the less-used paths. We have already developed tools to automatically collect the popular paths data for LinuxKit by running the top 100 popular Docker containers in it. Next, we designed and implemented a system to automatically instrument the Linux kernel based on our popular paths data. Our Secure Logging System inserted warning messages at the beginning of each basic block in the unpopular paths, and produced a Secure Logging Kernel. Our modified Linux kernel monitors the use of kernel paths by untrusted user applications, and will output warnings whenever there is an attempt to access the unpopular paths. Currently, we are testing to get the exact numbers of how many unpopular paths will be accessed when running popular Docker containers with our instrumented kernel. We expect to see very small percentage of unpopular paths used by popular containers.

Step three: future work

Our final step is to refine and extend our current system to fully automate the application of our popular paths metric, and make it easier to utilize our metric on multiple kernel versions. Currently, we mainly worked on the Linux kernel version 4.14.24. And we have also gathered the popular paths data, and instrumented the source code for the Linux kernel version 4.9.86. We want to make our tool be able to automatically work on multiple existing kernel versions and new releasing versions. Moreover, there are more than two million available Docker images on Docker Hub [4]. And we want to make it easy to automatically profiling the kernel trace and obtaining the popular paths data for these containers by using our system, so that it can provide large-scale dataset that may benefit our research community.

Conclusion

During my PhD study, the research problem that I have been focused on is that virtual machines and containers have security concerns of triggering zero-day bugs in a vulnerable host operating system kernel. To solve this problem, we first studied and developed a “popular paths” metric that revealed a strong correlation between the popularity of kernel code and the likelihood that it contains security bugs. We then designed and developed a system to apply our popular paths metric to the LinuxKit virtual machine. As our current on-going work, we have obtained the popular paths data for the LinuxKit VM, and instrumented the Linux kernel source code based on this data. We are now running our testing containers on the modified host kernel, and verifying that real-world popular Docker containers can run their default workload using the popular paths. Next, we will extend our current work to automate the application of our popular paths metric across multiple kernel versions. We also plan to automate the profiling of popular paths data on a large-scale dataset of containers. Our hope is that by the end of my PhD study, we will be able to demonstrate that the popular paths metric is an effective solution to enhancing container security by avoiding kernel vulnerabilities, and can be used in practice to benefit both security researchers and container users.

References

[1] 451 Research, Cloud-Enabling Technologies Market Monitor Report. "https://451research.com/images/Marketing/press_releases/Application-container-market-will-reach-2-7bn-in-2020_final_graphic.pdf", 2017.

[2] CVE-2016-5195, Dirty COW, Docker container escape. "https://blog.paranoidsoftware.com/dirty-cow-cve-2016-5195-docker-container-escape/", 2016.

[3] Docker, Enterprise Container Platform for High Velocity Innovation. "https://www.docker.com", 2019.

[4] Docker Hub, A Library and Community for Container Images. "https://hub.docker.com", 2019.

[5] Linux kernel vulnerabilities from CVE Details. "https://www.cvedetails.com/product/47/Linux-Linux-Kernel.html?vendor_id=33", 2019.

[6] Li, Y., Dolan-Gavitt, B., Weber, S., and Cappos, J. Lock-in-pop: Securing privileged operating system kernels by keeping on the beaten path. In *Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC '17)* (Santa Clara, CA, USA, 2017), USENIX Association, pp. 1–13.

[7] Linuxkit, a toolkit for building secure, portable and lean operating systems for containers. "https://github.com/linuxkit/linuxkit", 2019.