MSc Security and Network Engineering

(SNE/OS3)

Research Project 1

Categorizing container escape methodologies in multi-tenant environments

by

Rik Janssen

Rik.Janssen@os3.nl
ID: 12000256

February 11, 2018

6 ECTS
January 8th - February 7th, 2018

Supervisor:
Max Hovens, KPMG Cyber Security Netherlands

Assessor:
Cees de Laat

University of Amsterdam

Graduate School of Informatics

Abstract—**With regards to operating-system-level virtualization, the security aspects that arise within multi-tenant environments utilizing this technology have not been systematically categorized. As a result of this omission, container escapes within such environments might occur. These escapes can lead to the loss of control and potential data leakage. This paper extends on previous work by providing additional security requirements and mitigation techniques. This contribution is based on an in-depth look at the architecture of the cluster management software enabling the aforementioned environments.**

Index Terms—**Containers, multi-tenancy, Kubernetes, operating-system-level virtualization, security.**

1 Introduction

With the continual emergence of operating-system-level virtualization the security aspects of containers become all the more important. Numerous vulnerabilities, such as [1]–[3], have been identified, which allow for container escapes to occur. These escapes can lead to compromising the host system itself and/or the reveal of confidential data. This is especially damaging in multi-tenant environments, because within such environments the same physical and/or logical resources (e.g., the processor, memory, file system, and management software) are shared between multiple potentially untrusted parties. A common example of these kind of multi-tenant environments is a public cloud. Categorizing the previously mentioned vulnerabilities remains a challenge, since not all vulnerabilities affect all container technologies and are also depended on specific configurations.

The aim of this paper is to systematically categorize these kinds of vulnerabilities, such that a systematic way of thinking can be achieved when handling the security aspects of multi-tenant environments (e.g., the aforementioned data leakage). This systematic way of thinking then allows for the specifying of mitigation recommendations.

This paper is further divided into six sections. Sections 2, 3, and 4 describe the research question, related work, and used methodology respectively. Sections 5 and 6 describe and discuss the results. The last section concludes this paper and lists the future work that can be performed.

2 Research question

The research question is stated below. Section 4 outlines the approach that was taken to answer this question.

- *How to systematically categorize vulnerabilities relating to multi-tenant environments that make use of operating-system-level virtualization?*

3 Related work

Multiple studies have been performed on the aspects of container security. [4] defined an attacker model and several security requirements for container technologies. [5] further looked at the mentioned attacker model, which it considered as an academic view, and unified it with an industrial view. [6] extended on the research done in [4] by categorizing how exploits of specific container security threats violated the security requirements.

[7] and [8] specifically targeted the `Docker` container platform. These studies covered the internals specific to Docker, but also the general technologies in the areas of compute, storage, and network that Docker makes use of (i.e. Linux kernel features), together with their related configurations. Comparisons between containers and virtual machines were also made by [9], and a hybrid approach of running containers within virtual machines is generally advised by the aforementioned studies. [10] took an extensive look at Docker as well as `LXC` and CoreOS's `Rkt`, and subsequently provided multiple security recommendations focusing on said technologies and overall containers in general.

In regards to the security of multi-tenant environments, [11] explored the design of 'automated threat mitigation architecture' using the `Kubernetes` cluster management software. The study described an event-driven process entailing the automated creation of secure images and container quarantine. Furthermore, in regards to secure images, [12] proposed a methodology to evaluate the security of container images.

The added value of the research proposed in this document is categorizing container vulnerabilities when moving the context from individual container host systems to cluster management software enabling multi-tenant environments.

4 Methodology

The conducted research consisted of a literature study only, no software was installed or infrastructure deployed. The focus of the research was to categorize existing vulnerabilities within multi-tenant environment, not to discover new vulnerabilities. Given the time constraints of the research, the research was limited to the Linux operating system and the Kubernetes cluster management software. These two were chosen based on their broad utilization as well as their history with Google, who has more than a decade of experience using these technologies as part of its `Borg` system, as stated in [13].

Only operating-system-level virtualization was a factor, other forms of virtualization (e.g. full or para) were not part of the research. Hybrid solutions utilizing both containers and virtual machines (e.g. projects like OpenStack's `Kata Containers project` or Xens `PV Calls`) did also not fall into scope; only container-native technologies were accounted for.

The research was divided into several steps. First, the cluster management software powering the multi-tenant environments was broken down into its components. Then possible areas of existing vulnerabilities, identified using the `Common Vulnerabilities and Exposures` (CVE) system, were mapped to these components. After this step was completed, the security requirements of container technologies, mentioned in the related works [4] and [6], were extended to account for the mapped vulnerabilities relating to the cluster management software. Furthermore, these vulnerabilities were associated with configurations in order to determine possible affected industries and workloads. Finally, several mitigation techniques were listed.

5 Results

The results of the literature study are presented in the subsections below.

5.1 Architecture overview

Figure 1 gives an overview of the scope of the `Cloud Native Computing Foundation` (CNCF).

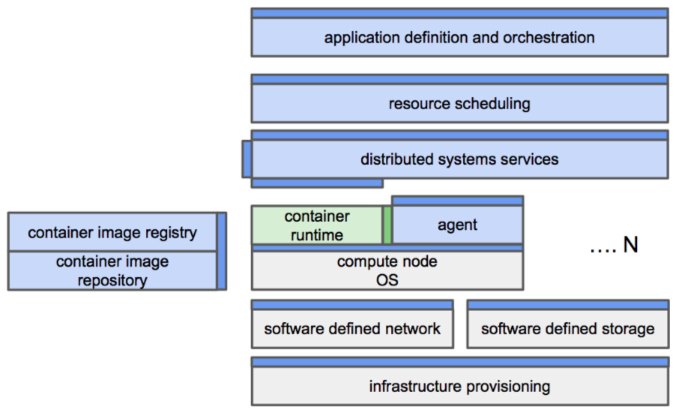

Figure 1. Scope of the Cloud Native Computing Foundation. Source: [14]

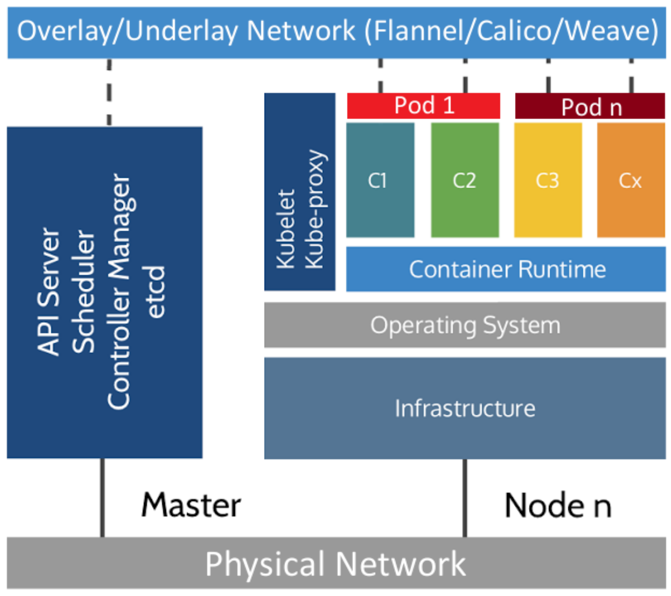

Figure 2. Architecture overview of Kubernetes. Adapted from: [17]

This foundation's mission is to "[...] create and drive the adoption of a new computing paradigm that is optimized for modern distributed systems environments capable of scaling to tens of thousands of self healing multi-tenant nodes."[15] Practically, the CNCF oversees multiple open source projects in order to enable the modern software delivery workflow (i.e., the implementation of the twelve-factor app methodology[16] in the form of microservices on top of container platforms la Docker). These project include Kubernetes for orchestration, `Prometheus` for monitoring, container runtimes like `containerd` and `rkt`, and many others. The CNCF's members include companies like Google (whom donated Kubernetes), Docker, Red Hat, Amazon, Cisco, Intel, IBM, Dell, VMware. and many others. The foundation is overseen by the Linux Foundation.

Figure 2 illustrates the architecture of the Kubernetes cluster management software [18]. Comparing this figure to the scope of the CNCF, it's clear Kubernetes materializes the CNCF's scope within its software architecture. The major area of interest related to this paper is the 'Master'. This node controls all the nodes of a cluster. Each node runs an operating system aimed at running containers. Kubernetes manages the containers as `pods`, which is a layer of abstraction referring

Hardened application
User namespace w/o caps
Mount protections
Minimal container distro
Syscall Filtering w/ seccomp-bpf
Linux kernel with grsecurity+pax
Hypervisor/Hardware

Figure 3. NCC Group's security model. Adapted from: [19]

to the tight-coupling of the containers (i.e., these containers, and corresponding applications, will be deployed on the same node). Furthermore, the component `etcd` is a distributed key-value store which functions as the single source of configuration truth (e.g., IP addresses, storage locations, security policies, etc.) of the entire cluster.

To clarify, Kubernetes itself is just the orchestration software (i.e, all the components that make up the master node plus the two components running on each node). The other components, such as the network and operating system, are managed by Kubernetes but can be implemented using different software (e.g., the network connecting the containers can be realized using software like `Flannel` or `Calico` which is then utilized by Kubernetes). In turn, the software responsible for providing this functionality to Kubernetes relies on further lower-level components like file systems (e.g., `overlayfs`) and kernel features (e.g., `cgroups` and `namespaces`). In other words, Kubernetes is not a complete solution in itself, but rather relies on other software and extends their functionality. This means a layered software model is utilized to enable multi-tenancy within environments built on operating-system-level virtualization.

Finally, Kubernetes utilizes the concept of `namespaces` in order to separate resources per tenant. These namespaces are separate domains of control within a Kubernetes cluster (i.e., resources can be restricted to certain namespaces). Please note that these namespaces are not the same as the earlier mentioned kernel features.

5.2 CVE mapping

Because Kubernetes is using a layered approach, one would have to map the CVEs of every single component and their subcomponents. For example, Kubernetes might utilize the Docker runtime, hence any vulnerabilities related to the Docker runtime now also impact the kubernetes cluster. Furthermore, any vulnerabilities related to the Linux kernel features might impact Docker. So mapping these CVEs would quickly encompass the entirety of the Linux software landscape. Figure 3 shows a security model provided by the NCC group that accounts for these layers.

In order to still account for the mapping of the CVEs, the CVEs relating to Red Hat's `OpenShift` platform have been looked at [20]. This platform combines a Linxu distribution, Docker, and Kubernetes. Currently there are 76 CVEs listed. As such, these CVEs encompass the aforementioned technologies, providing a holistic overview.

One major CVE of current note is the Meltdown vulnerability discussed in [21]. This hardware-level vulnerability makes use of out-of-order execution and CPU cache timings to read all physical memory from any user space process. The paper

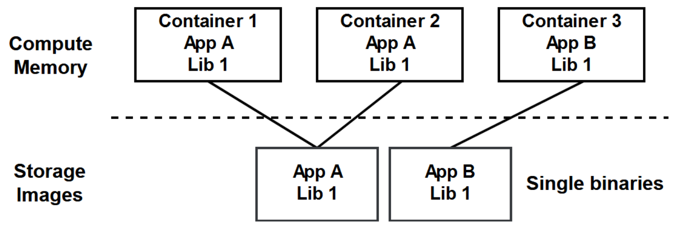

Figure 4. Static linking of binaries within an OS-level virtualization platform.

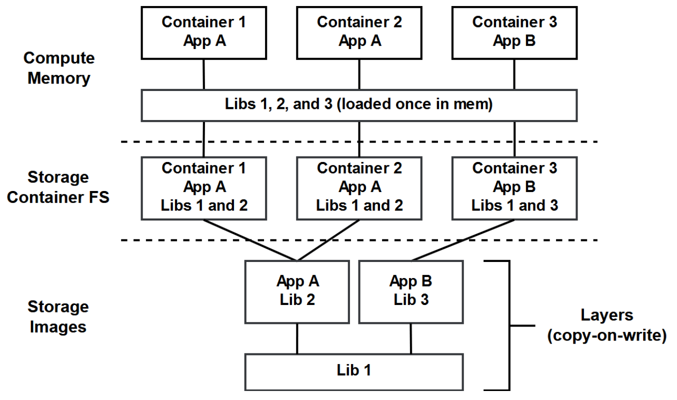

Figure 5. Dynamic linking of binaries within an OS-level virtualization platform.

specifically mentions the leakage of data between containers. As such, the boundary between tenants has moved from the visualization layer to the physical layer.

5.3 *Extended requirements*

Based on the previous described results, the following requirements are provided in order to supplement the ones defined in the previous done work done by [4] and [6]. The requirements are as follows:

- Separate security-sensitive workloads over physical nodes because the security boundary is moved to the infrastructure (i.e. hardware) level instead of the virtualization level.
- Secure the single source of configuration truth because this is a single point of failure within a multi-tenant environment.
- Utilize an integrated approach in regards to the cluster management software (i.e., don't use upstream Kubernetes unless processes are in place to track the individual components' vulnerabilities).

5.4 *Mitigations*

Figures 4 and 5 show a possible mitigation technique involving static linking and dynamic linking of binaries within containers. Specifically, figure 5 shows a way to optimize memory and storage usage.

6 Discussion

Looking at the results, it seems that operating-system-level virtualization by itself doesn't provide the required security boundaries for multi-tenant environments. Furthermore, the security aspects can be divided into two categories: infrastructure and application. Because applications are tightly-coupled to the underlying infrastructure when using operating-system-level visualization, hardening is required. Within multi-tenant environments consisting of untrusted parties this remains a challenge. Finally, the recent acquirement of CoreOS by Red Hat might show a movement to a single Enterprise-ready container platform.

7 Conclusion and future work

This paper provided a view of the container security landscape when accounting for multi-tenancy. As a result, several requirements and mitigation techniques have been identified, allowing for a systematic approach to the securing of multi-tenant environments utilizing operating-system-level virtualization.

Future work might include a correlation of the requirements to compliance standards such as ISO and HIPAA. Furthermore, other operating-system-level virtualization platforms and cluster management software could be looked at (e.g., `Microsoft Windows`, `FreeBSD`, `Apache Mesos`, `Nomad`, etc.). Other virtualization techniques (e.g. full and/or para) could also be relevant, especially the out-of-scope hybrid solutions that were previously mentioned in section 4. A further interesting study might be to look at the automation of the classification and orchestration of workloads (i.e. apply the requirements in a automated fashion). Finally, in regard to application architecture, a look at the `Istio` platform might be relevant. This open source project (provided by Google, IBM, and Lyft) is currently in alpha and aims to provide secure microservices on top of Kubernetes.

References

[1] CVE-2016-5195. Available from MITRE, CVE-ID CVE-2016-5195. 2016. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5195. [Accessed: Jan. 22, 2018].

[2] CVE-2015-3630. Available from MITRE, CVE-ID CVE-2015-3630. 2015. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3630. [Accessed: Jan. 22, 2018].

[3] CVE-2017-5123. Available from Red Hat, CVE-ID CVE-2017-5123. 2017. [Online]. Available: https://access.redhat.com/security/cve/cve-2017-5123. [Accessed: Jan. 22, 2018].

[4] E. Reshetova, J. Karhunen, T. Nyman, and N. Asokan, "Security of os-level virtualization technologies," in *Nordic Conference on Secure IT Systems*. Springer, 2014, pp. 77–93, doi:10.1007/978-3-319-11599-3_5.

[5] L. Catuogno and C. Galdi, "On the evaluation of security properties of containerized systems," in *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, Dec 2016, pp. 69–76, doi:10.1109/IUCC-CSS.2016.018.

[6] S. Laurén, M. R. Memarian, M. Conti, and V. Leppänen, "Analysis of security in modern container platforms," in *Research Advances in Cloud Computing*. Springer, 2017, pp. 351–369, doi:10.1007/978-981-10-5026-8_14.

[7] T. Bui, "Analysis of docker security," *arXiv preprint arXiv:1501.02967*, 2015, [Online]. Available: https://arxiv.org/abs/1501.02967v1. [Accessed: Jan. 8, 2018].

[8] A. R. MP, A. Kumar, S. J. Pai, and A. Gopal, "Enhancing security of docker using linux hardening techniques," in *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, July 2016, pp. 94–99, doi:10.1109/ICATCCT.2016.7911971.

[9] M. Eder, "Hypervisor-vs. container-based virtualization," *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, vol. 1, 2016, [Online]. Available: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-07-1/NET-2016-07-1_01.pdf. [Accessed: Jan. 8, 2018].

[10] A. Grattafiori, "Understanding and hardening linux containers," *Whitepaper, NCC Group*, 2016, [Online]. Available: https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/2016/april/ncc_group_understanding_hardening_linux_containers-1-1.pdf. [Accessed: Jan. 8, 2018].

[11] N. Bila, P. Dettori, A. Kanso, Y. Watanabe, and A. Youssef, “Leveraging the serverless architecture for securing linux containers,” in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, June 2017, pp. 401–404, doi:10.1109/ICDCSW.2017.66.

[12] B. M. Abbott, “A security evaluation methodology for container images,” 2017, [Online]. Available: https://scholarsarchive.byu.edu/etd/6287. [Accessed: Jan. 17, 2018].

[13] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, “Large-scale cluster management at google with borg,” in *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015, p. 18, doi:10.1145/2741948.2741964.

[14] The Linux Foundation. Cloud Native Computing Foundation (CNCF) Charter. 2017. [Online]. Available: https://www.cncf.io/about/charter. [Accessed: Jan. 15, 2018].

[15] T. L. Foundation, “Cloud Native Computing Foundation,” 2017, [Online]. Available: https://www.cncf.io/. [Accessed: Jan. 17, 2018].

[16] A. Wiggins, “The Twelve-Factor App,” 2017, [Online]. Available: https://12factor.net/. [Accessed: Jan. 17, 2018].

[17] I. Gunaratne, “A Reference Architecture for Deploying WSO2 Middleware on Kubernetes,” Jan. 2017, [Online]. Available: https://medium.com/containermind/a-reference-architecture-for-deploying-wso2-middleware-on-kubernetes-d4dee7601e8e. [Accessed: Jan. 17, 2018].

[18] T. K. Authors, “Concepts — Kubernetes,” 2018, [Online]. Available: https://kubernetes.io/docs/concepts. [Accessed: Jan. 17, 2018].

[19] A. Grattafiori. Def con 23 - aaron grattafiori - linux containers: Future or fantasy? - 101 track. Youtube. 2015. Available: https://www.youtube.com/watch?v=iN6QbszB1R8 [Accessed: Jan. 3, 2018].

[20] MITRE, “Redhat Openshift : List of security vulnerabilities,” 2018, [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-25/product_id-23704/Redhat-Openshift.html. [Accessed: Jan. 31, 2018].

[21] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, “Meltdown,” Jan. 2018, [Online]. Available: https://arxiv.org/abs/1801.01207. [Accessed: Jan. 30, 2018].