Automatic Security Patching for Containerized Java Applications

Olufogorehan Tunde-Onadele
oatundeo@ncsu.edu
North Carolina State University
Raleigh, North Carolina

Xiaohui Gu
xgu@ncsu.edu
North Carolina State University
Raleigh, North Carolina

ABSTRACT

Containers have become increasingly popular in distributed computing environments. However, recent studies have shown that containerized applications are susceptible to various security attacks. We need to address underlying software vulnerabilities in these applications. Existing research has attempted to analyze the underlying software bugs with various static and dynamic approaches to extract vulnerability patterns in the application. However, these patterns can vary among applications and libraries. In addition, many solutions do not address the operational constraints and objectives of distributed infrastructures. In this work, we explore static program analysis to find and understand unique code patterns in the core Java libraries that are vulnerable to security attacks. Thereafter, we study how to deliver appropriate patches to Java applications in distributed container clusters.

CCS CONCEPTS

• **Security and privacy** → **Virtualization and security**.

KEYWORDS

Container Security, Program Analysis, Security Patching

1 INTRODUCTION

Containers have become increasingly popular in distributed computing environments because they provide an efficient and lightweight deployment method for various applications. These advantages have made containers the subject of recent research. However, studies [3][5] have shown that containers are prone to various security attacks, which has become one of the top concerns for users to fully adopt container technology [1].

Containerized applications also pose a set of new security challenges to distributed computing environments such as conventional computing clouds and data centers that use virtual machines. First, container image repositories are prone to vulnerabilities. A previous study [5] reveals an alarming degree of vulnerability exposure and spread in the official Docker Hub container repository. It is complex to maintain a public or private container repository which often consists of a large number of container images and many inheritance layers. If a container is created from a base image, any vulnerability detected in the base image needs to be patched in the containers that are built on top of the base image. Second, containers are often allocated with limited resources because a large number of containers often share the resources of a single physical host. Security patching causes significant resource increase (e.g., memory bloating) in a patched container, which could make the container unable to run after patching.

In addition, existing software repair fixes underlying software bugs using various static and dynamic approaches to extract vulnerability patterns in the application [2, 4, 6–8]. However, if these patterns are based on application code external to the core libraries of the programming language, the system may not detect similar issues in other applications and libraries because of differences among their code structures.

In this work, we explore static program analysis to find and understand unique code patterns in the core Java libraries that are vulnerable to security attacks. Since applications that build on these key libraries will eventually call the underlying library functions, such analysis can apply to diverse applications. After we extract such patterns, we study how to deliver suitable patches to Java applications in distributed container clusters. We conduct this study on real world security bugs in popular Java server applications.

REFERENCES

[1] Anthony Bettini. 2015. Vulnerability Exploitation in Docker Container Environments. https://www.blackhat.com/docs/eu-15/materials/eu-15-Bettini-Vulnerability-Exploitation-In-Docker-Container-Environments-wp.pdf.
[2] Ting Dai, Jingzhu He, Xiaohui Gu, Shan Lu, and Peipei Wang. 2018. Dscope: Detecting real-world data corruption hang bugs in cloud server systems. In *Proceedings of the ACM Symposium on Cloud Computing*. 313–325.
[3] Docker Image Vulnerability Research. 2017. https://www.federacy.com/docker_image_vulnerabilities.
[4] Jingzhu He, Ting Dai, Xiaohui Gu, and Guoliang Jin. 2020. HangFix: automatically fixing software hang bugs for production cloud systems. In *Proceedings of the 11th ACM Symposium on Cloud Computing*. 344–357.
[5] Rui Shu, Xiaohui Gu, and William Enck. 2017. A Study of Security Vulnerabilities on Docker Hub. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. ACM, 269–280.
[6] Julian Thomé, Lwin Khin Shar, Domenico Bianculli, and Lionel Briand. 2017. Search-driven string constraint solving for vulnerability detection. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 198–208.
[7] Zhengzi Xu, Bihuan Chen, Mahinthan Chandramohan, Yang Liu, and Fu Song. 2017. SPAIN: security patch analysis for binaries towards understanding the pain and pills. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 462–472.
[8] Yunhui Zheng and Xiangyu Zhang. 2013. Path sensitive static analysis of web applications for remote code execution vulnerability detection. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 652–661.