Kubernetes-Based Environmental Security Analysis

Finlay Reid

BSc (Hons) Ethical Hacking

Introduction

Kubernetes is a container management software developed by Google, assisting in deploying and orchestrating container virtualization technology. Utilised by many leading organisations, the benefits are abundant, but concerns have been raised over how secure the software is.

In order to assist users in securing their Kubernetes environment, various tools have been developed—these range from remote testing services to static code analysers of configuration files.

This project analysed the methods by which security practitioners secure Kubernetes environments. Two practical elements were undertaken throughout project development, including an experiment on Kubernetes static analysis tools and the creation of a Kubernetes hardening script.

Aim

The following project aimed to complete a meticulous examination of Kubernetes-based environmental security, analyse different static analysis tools and develop a competent Kubernetes hardening script.

Objectives –

- Development of a Kubernetes hardening script that performs checks on several different aspects of a Kubernetes environment and presents the results in a readable format.
- Identify and investigate existing research related to Kubernetes best practices and the state of Kubernetes/container security.
- Conduct an experiment measuring the efficiency and resource usage of prevalent Kubernetes static analysis tools.

Method

The steps undertaken in the methodology, in chronological order were:

- Research of Kubernetes set up, static analysis tools and script information.
- Set-Up of the testing environment.
- Installation of Kubernetes static analysis tools.
- Testing of Kubernetes static analysis tools.
- Development of Kubernetes hardening script.
- Documentation of experiment and script development.

The research section involved compiling several current Kubernetes security auditing tools and investigating the setup method. The setup of the locally provisioned Kubernetes cluster enabled the installation of Kubernetes static analysis tools. Online documentation was followed throughout, and five prevalent static analysis tools were chosen based on the information discovered from the research phase.

The static analysis tools investigated included Checkov, Kubelinter, Kubeaudit, Kubescore and Kubesec. In order to test these tools to determine what would be the most suitable for the hardening script, the GNU time utility was employed to measure the resource usage and time taken for each of the tools. This process was accomplished by scanning the Kubernetes API YAML file using each security utility.

Finally, tools were tested five times to ensure an accurate representation of efficiency was determined.

Script development was undertaken in python and aimed to diagnose and implement fixes to Kubernetes environments. Developing the Kubernetes hardening script was iterative, with the script's file structure based on Kubernetes components.

Results

Testing of the Static Analysis Tools occurred pre-development of the script due to utilising the static analysis tool within the hardening script.

Checkov recorded a high of 12.21 seconds in user time and a low of 2.33. The results remained similar apart from the outlier recorded at the start of the experiment.

Kubeaudit was the third tool measured; it shared similarities with the results recorded when testing Kubelinter. Its system times were consistent in every test at 0.01, and its highest recorded user time topped out at 0.04.

Kubesec results were consistent for every metric with no outliers. The only notable aspect of this tool's results was the extremely low CPU usage throughout testing, with the highest at 16 and the lowest at 8.

The developed Kubernetes hardening script comprises several files that provide varying features and advantages to the user. The tool performs the following checks:

- File Permission
- File Ownership
- Network
- Authorisation (Checkov)
- Admission Controller (Checkov)
- Container Security Context (Checkov)

The tool can output the results in three formats. All of these have varying layouts and comprise different information. The tools outputs include :

- PDF Report Document
- CLI
- HTML Report

Discussion

Each of the Kubernetes security tools tested had its positives and negatives.

Checkov was the most suitable tool for the needs of the Kubernetes hardening script. Based on factors, including the familiarity with the language and the range of checks it performed on the Kubernetes configuration files. Furthermore, the result metrics throughout testing were steady, and a wealth of helpful documentation exists.

The decision to develop the script in python and structure the files by Kubernetes components was correct. Best practices in developing program applications were followed. However, several lousy code practices are littered throughout the script.

The tool developed in this project builds on the progress made by other Kubernetes security tools.

Conclusion

This project determined that Kubernetes security is exceptionally fragile in some aspects. However, Kubernetes security tools can be improved with the checks they perform and the user experience. Checkov was by some distance the most effective tool but not the most efficient; that title belonged to Kubescore.

References

Mytilinakis, P., 2020. *Attack methods and defenses on Kubernetes* (Master's thesis, Πανεπιστήμιο Πειραιώς).

Shamim, M.S.I., Bhuiyan, F.A. and Rahman, A., 2020, September. XI Commandments of Kubernetes Security: A Systematization of Knowledge