INTERNATIONAL JOURNAL OF ADVANCED INNOVATIVE TECHNOLOGY IN ENGINEERING

Published by Global Advanced Research Publication House Journal

Home page: www.ijaite.co.in

To Improve User and Architecture Part of Competitive Programming Website using Docker

[1]Shreyas Sable, [2]Saujanya Deshmukh, [3]Parag Lakade, [4]Sachi Katkar, [5]Priti Mujbaile, [6]Shiwani Mahure, [7]Dr. S. W. Mohod

[1,2,3,4,5,6,7]Computer Engineering, Bapurao Deshmukh College of Engineering Sevagram, Wardha, Maharashtra, India

[1]shreyas.sable2166@gmail.com, [2]saujanyadeshmukh123@gmail.com, [3]paraglakade03@gmail.com, [4]sachikatkar09@gmail.com, [5]pritimujbaile84@gmail.com, [4]shivanimahure251@gmail.com, [7]sudhirwamanrao@gmail.com.

Article History

Received on: 10 Feb. 2025
Revised on: 28 Feb. 2025
Accepted on: 30 March 2025

Keywords: Cloud Computing, Remote Code Execution, Sandbox Environment, Cyber Security.

e-ISSN: 2455-6491

DOI: 10.5281/zenodo.15396473

Production and hosted by
www.garph.org

ABSTRACT

With advancements in Web Technologies, a huge number of activities have transitioned from the local environment to the cloud environment. Cloud environment offers an easy-to-use solution to the user that does not require any setup, infrastructure and maintenance. One example of such a transition is Remote Code Execution (RCE). Prior to this, every system was required to have various language compilers/interpreters installed with their tooling. Remote code execution has allowed users to directly run code on cloud environments and obtain results. This saves them the effort, disk space and compute power. Widely used examples of such are Online Compilers. Although these cloud environments have provided convenience, they are often vulnerable to malicious code. Such code can often cause harm such as fork bombs, DDoS attacks, data theft and destruction to the server hosting the cloud environment. Implementing safeguards against such attacks are cumbersome, maintained manually and work only for well-known and predicted attacks. We propose a system where remote code execution occurs inside a sandbox environment. A sandbox environment is a safe file system where the user can perform actions without affecting the host system. Once the attack is done, the affected sandbox is replaced with a new healthy sandbox.

1. INTRODUCTION

During the process of remote code execution, the client is usually a web frontend that utilizes an API call with all the relevant information. This API call is usually a HTTP POST request that goes to the server hosting the cloud environment. The cloud environment then evaluates the HTTP POST request and sends back a response. Many modern tools and technologies use the JavaScript Object Notation (JSON) format. The response is usually the result of the evaluation. In this case the request is the program to be executed and the response is the output of the program. Since the evaluation happens on the server side. The users neither have to install any tool locally nor do they have to use their system's resources for the evaluation process.

A. Utilizing Docker

The cloud environment is vulnerable to malicious code. A user might write some code that deletes important system files, uses up all of the system's resources and many more, all of which can damage the server-side software and lead to server crash. Docker can be utilized to solve this problem. Docker is a containerization software widely used in industry. Docker can be used to create a lightweight virtual file system along with system tools that exist separately from the host system. This virtual environment is called a Docker Container. Since a Docker Container exists separately with its own file system, tools and resource allocation, any changes either constructive or destructive made to the Docker Container does not affect the host system in any way. Therefore, when the cloud environment receives code from the user, it can directly run it inside a Docker Container [2]. In case of malicious code, the Docker Container acts as a shield and protects the host system as it exists separately from it. Thus, the damage is contained to the container itself. Each Docker Container is reproducible using a set of instructions provided in a text file called a Dockerfile. This damaged Docker Container can then be replaced with a new healthy container ready for code evaluation.

B. Introduction to Competitive Programming

Competitive Programming [3] is a programming sport where the players receive a problem statement and are required to solve it in the most efficient manner. Competitive programming websites such as LeetCode, CodeChef and HackerRank provide a web-based editor and cloud based remote code execution environment. This leaves a possibility of malicious code attack on the server. We propose developing and coupling a competitive programming website with our own remote code execution software based on Docker to provide a complete solution.

2. OBJECTIVES

The system aims to fulfil the following key objectives:

- To create a well featured Docker Container: The Docker Container, which acts as the sandbox environment must have all the latest interpreters and compilers installed so that it can cater to every user's need. This requires a well written Docker file and proper initial setup.
- To develop an API system for client server communication: The process of communication between the user and cloud environment happens over the REST API in the request response manner. The system must have a well-developed user-friendly API to make this communication clear and capable of integrating it into other applications.
- To develop a user oriented competitive programming website: The system is best utilized and demonstrated via the development of a competitive programming website which leverages remote code execution heavily.

3. METHODOLOGY

Majority of the operations in the pipeline rely on REST APIs. The client makes a HTTP POST request with the payload containing the source code along with metadata such as the compiler/interpreter to be used for evaluation, extension of the program, command line arguments and standard input. The entire process consists of two parts: Request-Response mechanism associated with Client and Cloud and Code execution inside Docker container. Furthermore, the user-oriented side of a competitive programming website can also be improved to fulfill India's academic and professional requirements.

A. Request-Response mechanism associated with Client and Cloud

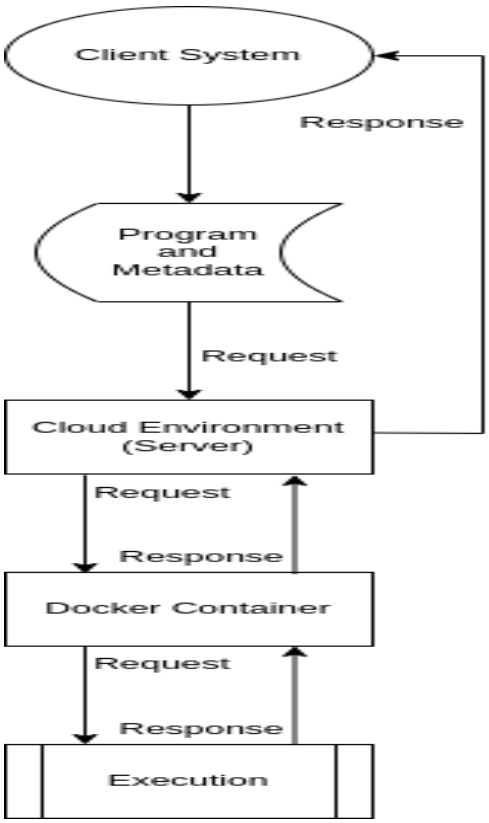

Figure 1: Representation of communication between client, cloud environment and the Docker container

Receiving Request: The client sends a program and metadata to the cloud environment via HTTP POST request.
Spawning Docker Container: A new Docker container is created via a child process. This Docker container has a server listening on its own.
Code Execution: The Docker container executes code and returns the output. This process is explained in the next section.
Container Replacement: After execution, the Docker container is removed and a new one is created to take its place. This process happens even if the container was not affected at all. It ensures security.
Sending Response: The result of the execution is sent back to the client.

B. Code Execution inside Docker container

Receiving Request: The Docker container receives the program and its metadata from the cloud environment as a HTTP POST request.
Disk Persistence: The program is written to disk in the form of a text file according to its metadata.
Execution: The server spawns a child process to execute the program using the appropriate language interpreter. During this procedure, command line arguments and standard input are also specified. In case of compiled programs, an extra child process is spawned prior to this process to carry out the compilation.
Output Generation: The server collects the output of the execution after the child process ends.
Sending response: The output is wrapped inside a JSON object and sent back to the cloud environment as a response. In case the program or the child process encounters any error, the error is reported as a response instead of the output.

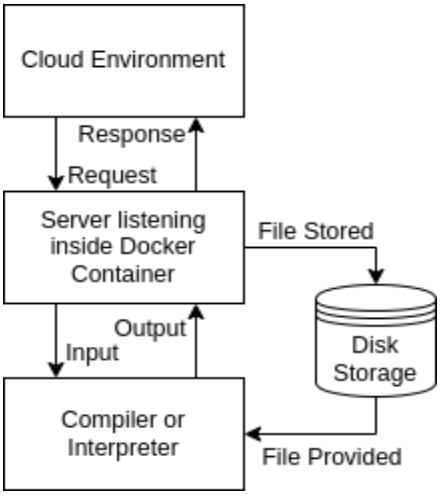

Figure 2: Representation of execution operations inside the Docker Container

C. Improvement on user side of Competitive Programming Website

Question formation and difficulty: Every segment of learners demand different problems for solving. These problems come from all points of the spectrum from easy to beginners. Although existing competitive programming websites offer this range, it is not always beginner friendly. A fresh take on competitive programming websites can address this issue.
Addition of more questions: Existing competitive programming websites have strict guidelines when it comes to problem submission. A new approach can enable users to form groups to add questions and share it within themselves without having to go through the extensive submission process.
Website as an exam platform: Educational institutes still rely on pen and paper for conducting programming exams. Hand written code is often confusing, hard to change and misleading. Secure Remote Code execution in the form of a website can be helpful in fixing this issue.
Website as a RCE platform: Many education institutes lack access to the latest software. Updating the software across a large number of systems requires a lot of effort and setup along with limiting factors from the old hardware. Secure Remote Code Execution via a frontend system like a competitive programming website can deal with this issue when hosted on a network.

CHALLENGES AND FUTURE SCOPE

A. Challenges

Resources Constraints: The docker container may reach a resource bottleneck given its resources if a computationally heavy code is executed inside it.
Load Distribution: The server might not be able to manage the incoming requests and the spawning of new docker containers in parallel leading to server crashes.
Latency: Users accessing the server that are physically away might face a delay in receiving output of their code in comparison to running it locally.

B. Future Scope

Kubernetes can be used for load balancing and addressing the issues with server crashes.

gRPC can be utilised instead of REST API to make the request-response mechanism faster. This addressed the latency issue.

AI can be integrated for automatic question generation to remove the minimal required human effort.

CONCLUSION

Advancements in cloud computing have made our lives easier. One such observation is Remote Code Execution and its convenience. However, there is a trade-off between convenience and security. Traditional methods of security that include observing attack patterns and implementing safeguard mechanisms are limited, manually maintained and too predictable for attackers. Hence, the usage of sandboxed environments can be used to contain these attacks instead of preventing them. This solves a huge problem for cyber security engineers and they do not need to worry about every edge case that an attacker can take advantage of. This paper and system demonstrate an alternative that is proven to be effective with a very little overhead in the guise of a competitive programming website. Furthermore, this system can be extended to fit in any other use case that requires secure remote code execution via REST API.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

FUNDING SUPPORT

The authors declare that they have no funding support for this study.

REFERENCES

[1] **S. Biswas, M. M. H. K. Sajal, and others, "A Study on Remote Code Execution Vulnerability in Web Applications," in Proceedings of the International Conference on Cyber Security and Computer Science, 2018.**

[2] **N. D. Kamod and R. N. Jadhav, "Secure and Scalable System for Online Code Execution and Evaluation using Containerization and Kubernetes," JETIR Journal, vol. 10, no. 2, pp. 1483-1490, Feb. 2023.**

[3] **P. Ribeiro and P. Guerreiro, "Early introduction of competitive programming," Institute of Mathematics and Informatics, Vilnius, 2007.6**