Practical Assignment #2

Container Management Tool

May 7, 2024

1 Introduction

As discussed in classes, Linux *chroot* is a powerful utility that allows changing the root directory of a process and its children to a new location in the file system. This allows the creation of a sandboxed environment where the process sees only the files and directories within the new root directory as if they were at the root of the filesystem. *Chroot* provides a level of isolation, allowing to run processes with restricted access to the rest of the system, which can help prevent unintended modifications or access to critical system files. *Chroot* can be used together with Linux namespaces and cgroups to create a lightweight, isolated environment for running applications.

Linux *namespaces* are a key feature of the Linux kernel that enables the creation of isolated environments within a single system. They provide a way to virtualize system resources such as process IDs, network interfaces, file system mounts, and more, thus enabling the creation of lightweight and portable environments (containers) that can run applications with their own view of the system.

However, because in contemporary computing environments, multiple processes compete for system resources like CPU, memory, disk I/O, and network bandwidth, it is necessary to allocate and manage these resources effectively, ensuring their fair distribution and preventing resource contention among applications. This is where Linux control groups, or *cgroups*, enter the "picture". *Cgroups* are a kernel feature that enables fine-grained control and resource management, allowing administrators to allocate system resources among processes (or groups of processes), ensuring that no single process can monopolize system resources to the detriment of others. This is particularly useful in multi-tenant environments, virtualized systems, or any scenario where resource contention needs to be managed effectively. With *cgroups*, administrators can enforce resource limits, prioritize workloads, and optimize resource usage.

Combining Linux containers (LXC) with *namespaces*, *chroot*, and *cgroups* forms a powerful toolset for creating lightweight, isolated environments.

2 Work to be done

The work to be carried out in this second practical assignment is the development of a command line management tool that enables running user applications in an isolated environment (container) employing Linux Containers, the "change root" utility (or a variant), *namespaces* and *cgroups*.

By leveraging namespaces, LXC ensures that each container has its own isolated view of system resources such as process IDs, network interfaces, and file system mounts, preventing interference between containers. The use of *chroot* allows LXC to change the root directory for a container, limiting its access to only a subset of the file system and enhancing security by preventing unauthorized access to critical system files. Additionally, *cgroups* play a crucial role in resource management, enabling to enforce limits on CPU, memory, disk I/O, and network bandwidth for each container, ensuring fair resource allocation and preventing one container from monopolizing resources to the detriment of others.

The basic functionalities to be supported by the management tool are:

- Create and delete a container;
- Execute system commands on the container (e.g., listing files) and see the results;
- Copy program/file into the container;
- Establish connectivity with the container;
- Run user applications in the container.

The group is free to use any programming language for the implementation of the management tool. However, an automated process to replicate the build environment (e.g., virtual environment) is required as well as clear instructions to generate the executable.

The intended management tool must allow to define usage limits of system resources (e.g., CPU, memory, etc.). The group must enforce the limits within the isolated environment and provide a demonstration of its correct operation. To attain the highest grade mark, the ability to dynamically change the limits in runtime also needs to be demonstrated.

The command line management tool must offer arguments for the different supported functionalities and a helper option must also be available to allow the user to easily understand the tool, its features and usage. Additional options (e.g., activity logging) will be valued in the grade. However, the rationale needs to be convincingly justified in a security perspective.

3 Grading

The project is expected to be implemented by a group of up to 2 students, and **MUST** reside in a private repository in the github/detiuaveiro organization, using the Github Classroom functionality.

Delivery should consist of a git repository encompassing a *README.md* file, the *codebase* and associated *documentation*.

- **README.md** file: Contains the project author information, the project description and its structure. The key implemented features are also explained in detail in this document;

- **Codebase**: Contains the developed application(s), script(s), etc., with appropriate comments that help a programmer to understand the considered options;
- **Documentation**: Contains support information (images, tables, etc.) that improve the description of the project. Also, it **MUST** contain a 10 minutes (max) video demonstrating the operation of the developed management tool, including the enforcement of isolation (e.g., memory usage, PID visibility, etc.).

The use of existing code snippets, applications, or any other external functional element without proper acknowledgment is strictly forbidden. If any content lacking proper acknowledgment is found, the current rules regarding plagiarism will be followed.

3.1 Assignement variants

Students are encouraged to explore alternative projects within the scope of this second practical assignment as long the project focuses on developing a secure application based on an isolated execution environment. However, in this case, the project MUST employ technologies that have not been addressed in practice in classes (e.g., ARM TrustZone, TPMs, etc.).

In case a group wishes to explore this avenue, a detailed description of the project needs to be submitted for validation within a week of the date of this document's publication.