GETTING STARTED
Introduction to Containers

Why Container Security?

Over the last decade, largely because of the Docker open-source project, the interest in the linux container technology has exploded. Containers use operating system-level virtualization to isolate applications from one another. They behave in much the same way as virtual machines using much fewer resources since they share the Linux kernel.

Docker has enabled developers to rapidly build and share applications, and Kubernetes has enabled these applications to scale up and down dynamically. This has led to the widespread adoption of Docker in the industry.

However, like all complex software systems, the container ecosystem is also prone to misconfiguration, leading to vulnerabilities that can be exploited by malicious users. Hence, understanding container security and being able to audit a container environment is a critical skill all security professionals should possess.

Prerequisites

- Basic knowledge of computers and networking
- Familiarity with the Linux operating system
- Basic knowledge of Docker will be helpful

What will you learn in the Container Security labs?

The Container Security labs will teach you the basic concepts of container security and pentesting. You will learn how to use various tools and commands to identify and exploit the different components of the container ecosystem.

While the lab sections can be completed in any order, the recommended order would be:

Container Basics

This section explores the container management systems Docker and Podman. The labs also cover the low-level components of the Docker system e.g. containerd, runc. Beginners will learn how to perform the basic operations like pushing, pulling, creating and running containers.

Docker Microservices

The ease of deployment of Docker containers has led to increased interest in a microservice architecture. This is an approach in which an application is built as a set of loosely coupled, scalable, maintainable, independent services. For example, a WordPress blog on the LAMP stack can be deployed on 2 containers rather than all on one node (machine/container) - the first container will hold the Apache web server and PHP with the WordPress files and the second container will contain the MySQL database. This section covers attacks on applications that are deployed in the microservice architecture. The attacker will attack the web applications exposed to the internet, get a foothold on the first container, and then use it as a pivot to attack/explore the other containers.

Container Breakouts

The term “Container Breakout” refers to the event where a malicious or legitimate user is able to escape the container isolation and access resources (e.g. filesystem, processes, network interfaces) on the host machine. This section covers the different misconfigurations and excessive privileges that can be used to break out of the containers. The labs are based on an assumed breach approach which means that the lab starts when the attacker has already gained a command shell on the container.

A Docker host is a machine on which the Docker daemon and Docker containers run. Once the Docker host is compromised, the attacker can also access all the other containers running on it. This section covers Docker socket misconfigurations that can be exploited by attackers to perform privilege escalation to take over the docker host. Scenarios, where insecurely configured host management tools can be leveraged to compromise the host, are also covered.

Docker Registry

A Docker Registry is used to store and share Docker images among users and systems. Public repositories are mostly protected with authentication. However, in the case of private registries, the users/organizations usually rely on existing security boundaries to protect the registry. This category deals with insecure Docker registries and attacks that can be done on Docker infrastructure using insecure registries.

Docker Image Analysis

A Docker image is a file that is used to create a container. It contains applications/binaries and files that define the function and behavior of the corresponding containers. Docker images are built using instructions in a plain-text file known as a Dockerfile. This category covers the analysis of Docker Image layers and how to recover overwritten artifacts from Docker images.

Docker Forensics

The Docker ecosystem comprises different components i.e. images, containers, networks, checkpoints, etc. All of these components can be investigated to understand the chain of events in case of an incident. The labs in this section deal with the forensics aspects of Docker components.

Docker API Firewall

Docker supports third-party plugins that can be used to enforce custom restrictions on the Docker daemon API. An API firewall can be created by clubbing such plugins together. This is not to be confused with the network firewall that can be created using IPtables. The labs in this section deal with bypassing or evading the restrictions applied to the Docker daemon REST API using plugins.

Docker Tools

Docker has a rich community of developers, sysadmins and security professionals. This community promotes Docker usage by knowledge sharing, knowledge exchange and creating tools to perform different tasks. There are tools that help in simplifying the management of Docker environments and tools that help in keeping the environment secure. This category explores the different types of tools that are used for managing and securing Docker ecosystems.