happiest minds
The Mindful IT Company
Born **Digital** . Born **Agile**

Container Security

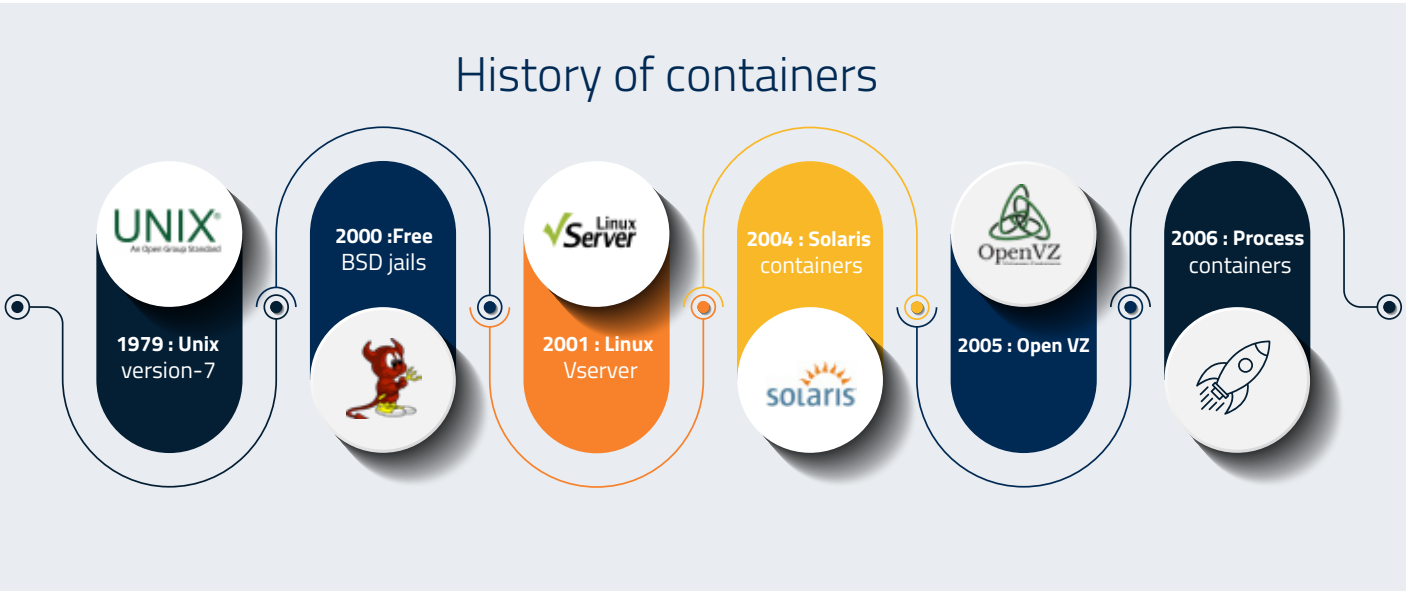
powered by DevSecOps

Meaning of Container

Containers are a type of software, which isolates environment running within a host machine's kernel that allows us to run application-specific code. Decoupling the containers-based applications can be deployed across an environment which can be either on the public/private cloud, personal laptop and data center as well.

Evolution of containers

Below timeline shows a brief history of containers



Unix V7 (1979)

Chroot system call was introduced in 1979 in Unix version -7. Changing the root (/) directory of process and corresponding child process to a new location in the file system. It was the beginning of process isolation, by parting file access to for each process.

FreeBSD Jails

BSD (Berkeley Software Distribution) incorporated chroot in 1982, FreeBSD Jails facilitated administrators to partition a FreeBSD system into several independent, chunks of systems called Jails. Also, FreeBSD Jails can be assigned with IP address for each system and configuration. Evolution of Jails ease administration which will isolate the service and customer security.

Linux VServer

Linux Vserver like FreeBSD Jails, partition resources, this can be implemented by patching the Linux kernel.

Solaris container

Solaris Containers constitutes system resource controls (file systems, network addresses, memory), and boundary separated by zones that leveraged features like snapshots and cloning of zone file system provided by zones.

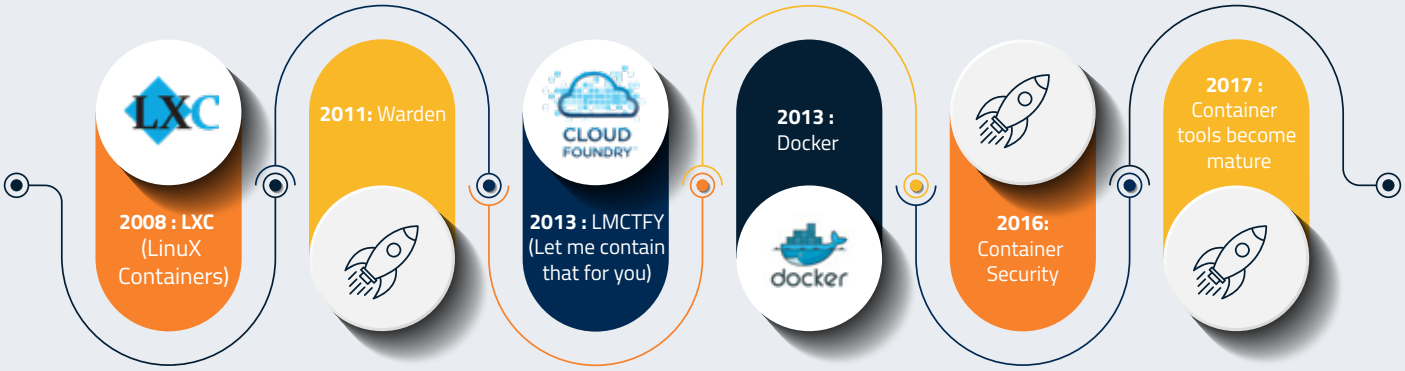
Open VZ

Open VZ is an OS-level virtualization technology for Linux which uses a patched Linux kernel for virtualization, isolation, resource management and check pointing.

Process Containers

Process Containers was designed for resource usage, accounting, limiting and isolating of a collection of processes. Later renamed to "Control Groups (Cgroups)" a year later and eventually merged to Linux kernel 2.6.24. Google launched the process container.

History of containers (Cont..)



LXC

It's the first and full implementation of Linux Container (LXC) manager without any patch requirement, and this was done using Cgroups and Linux namespaces which works on single Linux kernel.

Warden

Was started by CloudFoundry in 2011, using LXC in the early stages and later replaced it with its implementation. Warden can isolate environments on any OS, running as a daemon/service and providing an API for container management seeded the idea to develop the Docker container. It was developed as a client-server model to manage and build a collection of containers across multiple hosts, and Warden has a service to maintain Cgroups, namespaces and the process life cycle.

LMCTFY

Let Me Contain That For You (LMCTFY) was introduced in 2013 as an open-source version of Google's container stack, providing Linux application containers. Applications will be of Container aware, creating and managing their Sub-Containers. Deployment in LMCTFY stopped in 2015.

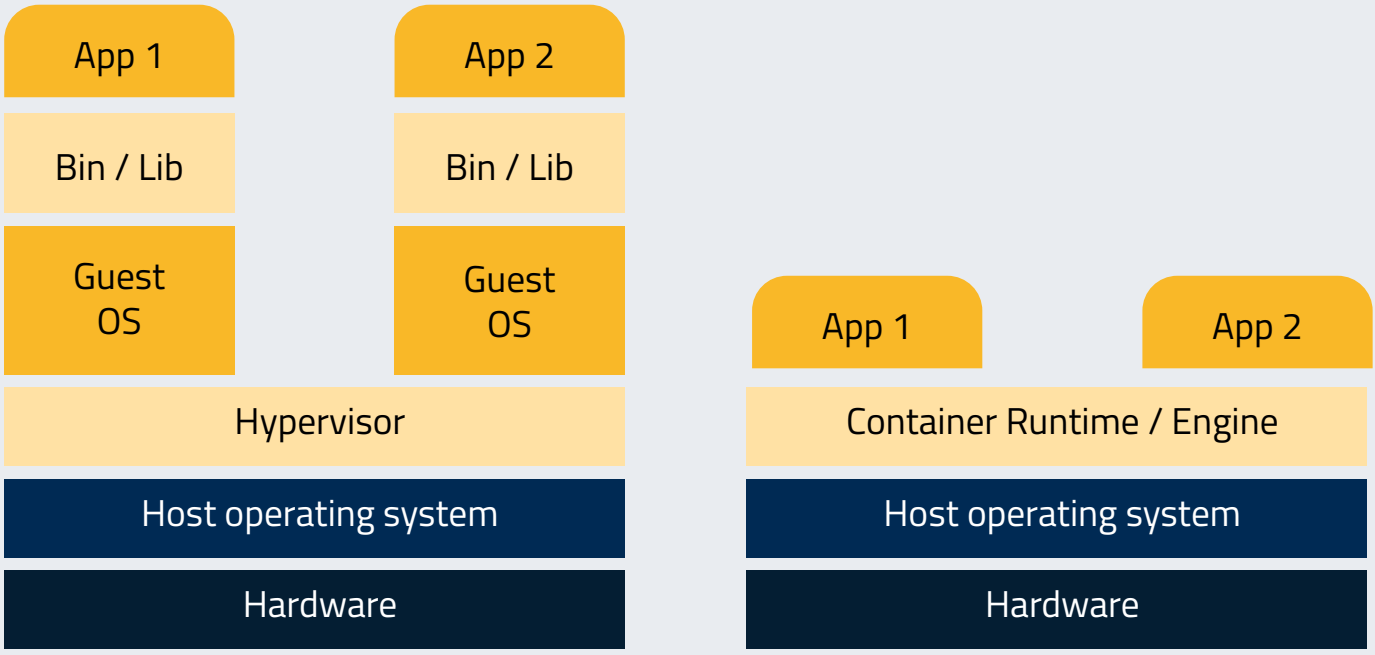
Docker

Popular containers at present across Enterprise, later 2013 has seen the growth of Docker as well in addition to LMCTFY, Same year container exploded in popularity. Like Warden, Docker also used LXC in its initial stages and later replaced that container manager with its own library, libcontainer. Docker offering an entire ecosystem for Container Management, separated itself from the pack.

Container for windows

Microsoft also launched its container support in 2016 called NANO for windows applications. Windows 2016 server will be shipped with the container as default support and Later included with windows 2010 as well. With this implementation, Docker would able to run Docker Containers in windows natively without any Virtual Machine to run Docker.

Containers Vs VM



Above diagram portray the difference between Virtualization and Containers, let us discuss how both works. Containers are different from Server Virtualization, and each Virtual Machine can run an OS (Which can be of Linux or Windows) in an independent environment and present to an application, via abstraction, a substitute to a physical machine.

The hypervisor isolates the Operating System and application from the underlying hardware. Hypervisor emulates hardware from pooled resources, which can be shared with multiple Virtual machines (Resources – Memory, CPU, Storage and Network).

In case of one VM-1, all the binary1/library1 and dependencies for application-1, If any another application-2 must be installed on same the VM-1 with different binary2/library2 and dependencies which might cause bottleneck situation for Application1. So, avoid this situation, and multiple machines for different application Containers will ease the administrator and developer job.

Like Virtual Machines, containers allow you to package your application together with libraries/binaries and other dependencies, providing isolated environments for running your software. Containers share the OS kernel, not the resources, only one instance of an OS can run many isolated containers. This feature allows containers to swap between OS; this feature adheres DevOps best practice.

The Importance of Container Security

With the extensive usage of Container-based applications, systems became more complicated on administrating and risk of increased security vulnerability for containers. Vulnerabilities like dirty COW only advanced the thinking. Along with the software development lifecycle it led to a deeper concern on security, making it a critical part of each stage in container app development, also known as DevSecOps.

Secure container with DevSecOps

All the above discussed are the history of containers and what makes differences between VM and container. In the sections below, let us understand the best practices and aspects as to how containers can be secured.

When deploying a container, it should be fortress by network policies, assigning roles (RBAC) and container image should be free from known vulnerabilities and have bare minimum functionalities.

According to "Cloud-native computing Foundation" Deployment of a container across the organization is taking off widely, most of the firms across the world have more than 250 containers deployed, as per stats 1 to 50 containers are deployed in less than a quarter. The increase in containers makes security more essential and more complicated to save at the same time.

Container image and Trusted/Signed

Layers of files are used to create Containers; the Container community calls these files as 'Container Images'. The most important for security is the base image because it is used as a checkpoint for spinning the container. While adding application, Package managers and while making config changes to containers, admins should be cautious enough and proactive, since we`re adding external content to an image.

Images shouldn't contain any secrets such as AWS or API keys. Developers sometimes leave Keys or secrets in images. It's advisable to supply the keys during runtime, statically embedding inside the image is inviting the attackers to exploit the containerized application.

Always prefer trusted/signed image from a registry, which provides another layer of protection. A good first step is to use a minimal base image as it is possible to run the application without any issue. Post successful application run, the image can be built with dev package, certify and trusted to make as vulnerability free template image for successive usage.

Containers should be immutable images with a minimal number of dependencies as they needed to run a single application in one container.

Tools for Containers

Different tools available in the marketplace and from open source enthusiast to scan the container image statically and dynamically to raise a flag when malicious or vulnerabilities are found, below are the popular software's

Twistlock: This software scan container for expected behavior and whitelist process, Networking setups like source-destination IP addresses, port and storage practices so that ambiguities are reported.

Aqua Security: Aqua software assists better practices for containers, Maintains the difference in container and originating image. Use to automate vulnerability protection and prevents CVE exploits.

Polyverse: This software will spin the container within a fraction of seconds and relaunch the application in a good state.

Container Patch Management

Patching containers isn't a good option though, and there can be of thousands of them in an environment irrespective of dev or pre-prod stage. Containers live very short – on an average two and a half-day if the organization is enabled with CI/CD process. However, if vulnerabilities detected in a container, it needs to be destroyed and rebuilt again from scratch to avoid the errors.

Container image has to components: The Base image and Application image. To patch a containerized system, we must update the Base image and then rebuild the Application image. In which case, therefore, the vulnerability in the Base image will be eliminated at an early stage and recreating the container in line with your typical CD process.

Runtime container security

By attaching Seccomp, Apparmor, or Selinux profiles to containers, you provide robust security isolation around the container to prevent it from making a SYSCALL.

Seccomp - system call operates on the **Seccomp** state of the calling process—this feature to restrict your application's access. **Seccomp** policies are defined using JSON files.

Apparmor – Its Linux security module used to protect the OS and its application from security threats, **Apparmor** profile blocks reading from unsafe directories.

Container Registry

A Container image is a kind of template that remains in storage, and each Container is an instance of an image as we store Container images and generate containers from those images as needed. There are several public registries and the repository in which container images stored is referred to as a registry.

DevOps pipelines to extended with scanners that search containers for packages and scan image with known vulnerabilities and alerts the owner if vulnerabilities detected in the Container.

Most of the organization maintain public registries such as Docker hub which provide their scanning service. For private registries, security providers such as "Twistlock" offers scanning capabilities. Keeping images in a private registry gives greater control and condition affecting its security. It's always advisable to use signed Container and pulling Container from public registry should be scanned statically for vulnerability and kernel threats before spinning the image. Using public registry for managing the COI isn`t wise, Private registry is precise for an enterprise solution. Owning an enterprise registry must be highly available, geographically replicated and ready for automation. Additionally, the private registry should have Log Management for auditing, Encrypted CLI, SSO and Vulnerability scanning.

The recent attack in Docker registry

"Docker Hub" has been hacked in April 2019 - https://threat-post.com/docer-hub-hack/144176/

The Recent researcher found 17 Crypto-mining containers in Docker hub.

Container Privilege

User privileges are always a top priority in a container which should be limited only perform the limited tasks. Providing elevated privilege to the user is inviting intruders to break out of the Container. Regular check on the default user privileges of any packages that you include in containers, and any open source container images that you include in your registry.

For example, Privileges should be kept close track while designing the containers for microservices. For example, when accessing the database, containers that read from or write into the database need the privileges required for those tasks. Never chose to allow users root access within a container.

Container daemon binds to Unix socket instead of TCP port, by default user root owns the Unix socket, and other users can access it using Sudo. Container daemon always runs as the root user. It's advisable to create a group and add users to it. When Container runs on privileged mode, Container can not only access to all hosts devices and files but also use most of host computer's kernel functions. You can use like Systemctl program or run docker daemon in a Docker container.

Container Networking Security

Securing containerized application is vital and more challenging since container-based application stack sits in a hybrid environment which needs to interact with both container and non-container application. To interlinking communication, we need to secure an L3 and L7 network for access and control management. Securing the rudimentary layer of L2 and L3 should be a priority between the container and non-container networks, whether it is facilitated by Internal firewall policies or routing tables, VPNs.

External connection to containers should be secured, ingress or egress should identify the vulnerability attacks. These policies shouldn`t be based only on an IP address. Containers can be constantly commissioned with different IP addresses for the same service (docker-compose in docker for providing service). A security policy with application protocol at Layer 7 can minimize error and automate the monitoring and security task, wrong protocol accessing or attempting to connect containers should be reported as a violation.

Author Bio

Sushena. P has 12 + years of experience in DevOps, Cloud/Data Center Automation, tools Integration, RPA, Architecture and Infrastructure solutions. He is responsible for DevOps/Automation practice strategy, solution and implementation across various projects. He is passionate on exploring and integrating products with cloud technologies.