Docker Security

Is it safe to run applications within containers?

Container Technology

- Linux Containers have been around for years (LXC)
- Docker made it really easy to use them
- The main focus of LXC wasn't security
- We should ask ourselves:
  - Is it even possible to run containers in a secure manner?
  - Are containers the perfect solution for every use case?
  - Do we need to rethink the way we deploy applications

Container Facts

- Container share the Kernel with the host
  - which is a huge difference to classic virtualization
- Kernel Namespaces are being used for separation
- cgroups are used for resource allocation
  - and prevent some kind of DDoS attacks
- Docker daemon needs root priviliges
  - a security vulnerability potentially affects all containers

Can Containers be made secure?

- Short answer: Yes
- But:
  - If you run an application as root, assume those apps can try anything to break out
  - suid can be used for privilege escalation
  - Kernel security issues might have impact on all Containers and the Host system
    - e.g. bogus syscalls like vmsplice() which has been discovered 2008
  - If UIDs are the same on the Host and inside the container, this might be used during attacks

Rule #1 – I am (not) root

- Always run regular applications as non-priviliged user inside a container
- Use capabilities for high level applications (but only if really necessary)
  - Suggestion: remove all Capabilities and only add those which are really required
    - e.g. you should normally not need to use route, ip etc in a container
- Drop capabilities if no longer needed
- CAP_SYS_ADMIN is a risk
  - Can literally do anything
    - If really required, use it during startup to set up services and then drop it

Rule #2 – If I am not root, whoami?

- User Namespaces
  - UIDs inside a container should be mapped to a different UID on the host
    - e.g. UID 0 in the container is mapped to some random UID outside the container
- Limits syscalls as they are done as unprivileged UID
  - Remember: x86_64 Linux kernel has over 600 system calls
- Caveat: permissions on mounted volumes might need to be adjusted

Rule #3 If root is really necessary, build walls

- In some rare cases root is necessary
- Run a VM inside a Container
  - to give it it's own Kernel
- Run your priviliged Containers within that VM
- Host ⟹ Container ⟹ VM ⟹ Container

Rule #4 Use Security Frameworks

- SELinux
- AppArmor
- Grsecurity
  - to harden the kernel
- seccomp-bpf
  - to whitelist/blacklist syscalls

Rule #5 readonly is your friend

- If ever possible run the container in readonly mode
  - easy to implement
    - Systemd tmpfiles.d
    - fedora-readonly.service
- write to noexec mounts only

Rule #6 only use trusted images

- If ever possible build your own images
- Run your own hub
  - e.g. using Satellite 6.x
- Use images from certified trusted sources only
    - <joeyh> I'll bet I could publish an image that just did a killall5 as root on startup and get plenty of people to nuke their container hosts
  - Red Hat is working on a Certification program
- Load images over trusted communication channels

Rule #7 update often

- Keep the Host OS up to date
  - Use live patching for the kernel
- Update your images regularly

Read On

- Docker Security
  - https://docs.docker.com/articles/security/
- Docker Security Future
  - https://opensource.com/business/15/3/docker-security-future
- Docker run Debian
  - https://joeyh.name/blog/entry/docker_run_debian/

Thank you for listening

and don't let the Whale Fail