redhat.

OPENSHIFT - CONTAINER SECURITY

How OpenShift applies Enterprise Security to the New World

Ian ‘Uther’ Lawson
Specialist Solution Architect

CONTAINERS DON’T CONTAIN!

ADDRESSING THE RISK - THE BASICS

Put simply, OpenShift addresses the security risks of the New Age of Containers by:

- Minimizing the attack potential (RHCO)
- Actually Containing the Containers (OpenShift Methods)
- Providing Edge-Sheep-Dipping (Quay/Clair)

RHCO?
“REDHAT COREOS”

As part of the acquisition of CoreOS by RedHat, the ContainerLinux platform will be offered as an option for hosting OpenShift on.

- Immutable OS
- Config injection using Ignition
- Tiny attack vector
- Nothing in the OS *other* than foundations for Container execution
- Release cadence tied into OpenShift Container Platform

UNCONTAINED CONTAINERS - THE RISKS

If using Containers ‘out of the box’ there are the following intrinsic risks:

- Container running as Host root
- Container running unconstrained in terms of system resources
- Container running unconstrained in terms of file systems
- Docker is security-agnostic out of the box
- Docker is a singular Daemon containing *all* Container functionality

IS ROOT REALLY A RISK?

- Linux supports effectively two types of process, priv and non-priv (UID 0 and UIC non-zero)
- Containers are file-system/processes with delusions of grandeur
- Containers are effectively slices of the OS and operate within that structure
- A Container running as UID 0 therefore has *all* priv capabilities

KERNEL 2.2+ ROOT CAPABILITIES

- CAP_SETPCAP (set process capabilities)
- CAP_SYS_MODULE (insert/remove Kernel modules)
- CAP_SYS_RAWIO (modify Kernel memory)
- CAP_SYS_PACCT (config process accounting)
- CAP_SYS_NICE (modify priority of processes)
- CAP_SYS_RESOURCE (override resource limits)
- CAP_SYS_TIME (modify system clock)
- CAP_SYS_TTY_CONFIG (config tty devices)
- CAP_AUDIT_WRITE (write the audit log)
- CAP_AUDIT_CONTROL (config audit subsystem)
- CAP_MAC_OVERRIDE (ignore Kernel MAC policy (!!!!!!))
- CAP_MAC_ADMIN (config MAC configuration)
- CAP_SYSLOG (modify Kernel printk behaviour)
- CAP_NET_ADMIN (config the network)
- CAP_SYS_ADMIN (all the other hairy bits of seriously messing up an OS)

HOW OPENSHIFT DEFENDS AGAINST ‘ROOT’

- OCP has removed all CAP_ capabilities from ‘root’ users running within a Container
- The user running within a Container is a non-zero UID within a ‘root’ group
- This group also makes the following mount points *read only* - /sys, /proc/sys, /proc/sysrg-trigger, /proc/irq and /proc/bus
- OCP uses ‘seccomp’ profiles to restrict system calls

KERNEL 3.8+ NAMESPACE CONTROLS

- OCP uses PIC Namespaces and Network Namespaces to control Containers
- These allow OCP to isolate process activities within a Container
- PID Namespace hides *all* non-namespaced processes from the running Container (if you ‘ps -ef’ in a Container you only see it’s relevant processes)
- Network Namespace allows OpenShift to impose network isolation on the Container

CGROUPS - RESOURCE CONTROL

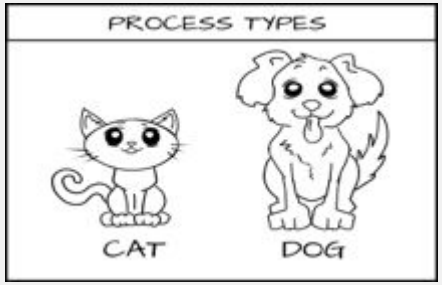
- Linux Kernel feature that imposes resource control upon a Process
- Limits can be strictly enforced, on a Container by Container basis, for CPU, memory, I/O and Network
- This negates any ‘noisy neighbour’ issues

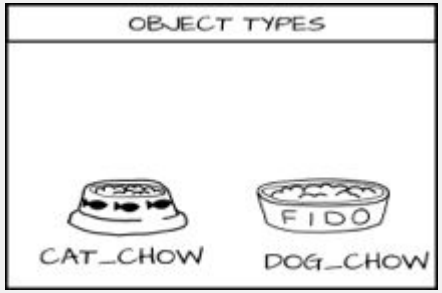
SELINUX - Security Enhanced Linux

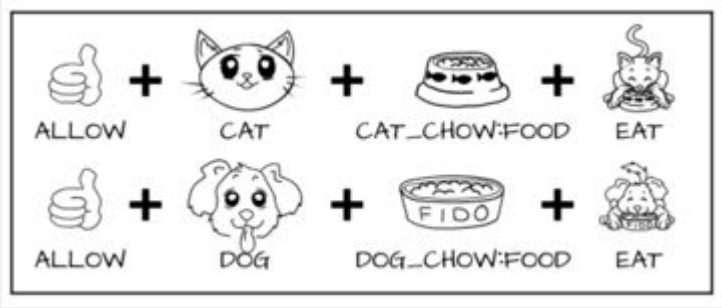
- Supports ‘Access Control Security Policies’
- Every Process and System Object gets labelled
- All Kernel operations are then labelled, classified and enforced using a set of defined rules.

An easier way to describe it fully is…….

SELINUX IN AMUSING PICTURES....

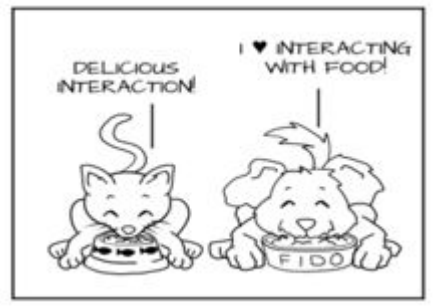






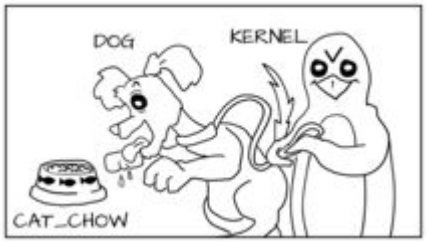
SELinux rules:
Allow cat cat_chow:food eat
Allow dog dog_chow:food eat

SELINUX IN AMUSING PICTURES....



The rules would allow the ‘cat’ process to eat ‘food’ labelled with ‘cat_chow’ and the ‘dog’ process to eat ‘food’ labelled with ‘dog_chow’

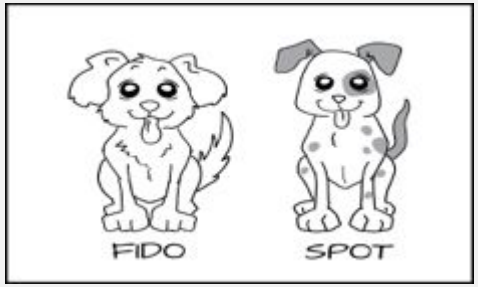
In an SELinux system *everything* is denied by default, so the Kernel will prevent the ‘dog’ process eating the ‘cat_chow’ ‘food’



This is called ‘Type Enforcement’ and protects the host system from Containers.

SELINUX IN AMUSING PICTURES....

To protect Containers from each other we use MCS, Multi-Category Security



This allows ‘sub-labelling’ of an object type, in this case ‘dog’. For example we could label ‘Fido’ as ‘dog:random1’ and ‘Spot’ as ‘dog:random2’

MCS rules say that *if* the Type Enforcement rules are OK *and* the random MCS labels match, access is allowed. Otherwise *denied* by default.

EDGE-SHEEP-DIPPING

- The OpenShift security methods protect the Container behaviour
- To protect against Application or Exploit security risks you use the “Edge Sheep Dipping” paradigm
- This entails interrogating and quarantining the Images of Containers at the point at which they enter the system as a whole.

REGISTRIES

- Container Images are immutable
- Once created (and stored) they don’t change
- Containers are instance of the Images with a thin writeable overlay file layer added
- The immutable Images are served from a Registry
- OpenShift maintains an Integrated Registry within itself, but can be fed by an external Registry for base Images and centralised control

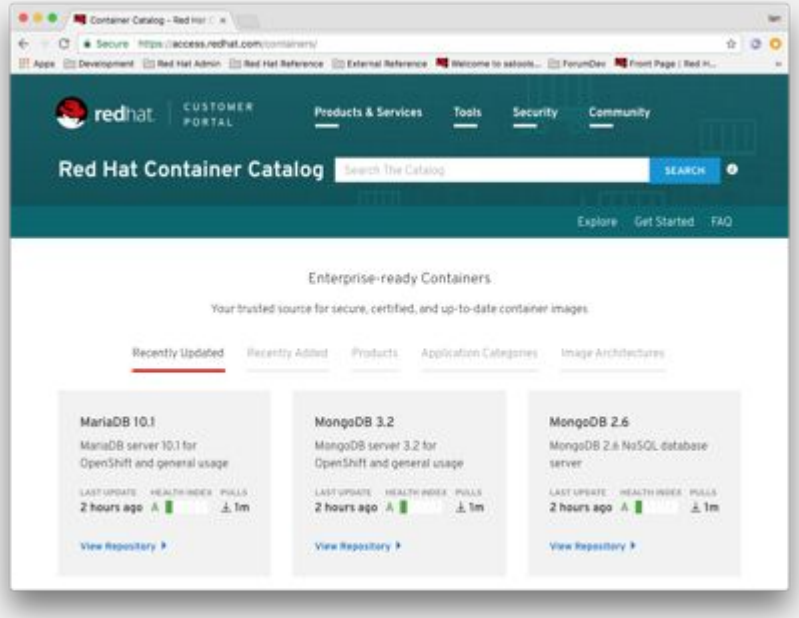
REDHAT QUAY

- As part of the CoreOS acquisition the Quay Registry product is now part of the RedHat product suite
- It offers a lot of advanced functionality around Registry
- Rollback, Federation, but most importantly exploit scanning
- This is done through a project called ‘Clair’
- This provides webhook driven events that indicate failure of a scan, which allows OCP to annotate and quarantine the Image for further action

REGISTRY SOURCE

- ‘Docker pull exploit_my_system’
- registry.access.redhat.com - secured and maintained RedHat base images
- OpenShift allows for blacklisting of upstream Registries and defaults to r.a.r.c

CONTAINER HEALTH RATING



- All Images hosted on r.a.r.c have the latest OpenSCAP OVAL profiles applied and are given a health rating
- This allows users to know what state the Containers are in

TO SUMMARISE:
OPENSHIFT CONTAINS CONTAINERS

- OpenShift incorporates at the OS level a number of security and defence mechanisms to harden and secure Containers
- OpenShift provides the tools to control ingest and consumption of Images in a highly secured manner

Additional Material

- “Ten Layers of Container Security whitepaper” - https://www.redhat.com/en/resources/container-security-openshift-cloud-devops-whitepaper
- More info on ‘Clair’ - https://coreos.com/clair/docs/latest/
- Securing OpenShift (3.11) - https://docs.openshift.com/container-platform/3.11/security/securing_container_platform.html

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos