**Contraintes et Privilèges trouvés pour les runtimes choisies**

\* Sources

| Runtime | Privilèges Nécessaires | Contraintes |
| --- | --- | --- |
| RunC en mode rootful | **Commande runc spec**  **Namespaces**  pid , network , ipc, cgroup , mount , uts  **Capabilités**  Bounding : "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE"  Effective (Effectifs Pour un processus Courant): "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE" Permitted : "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE"  **User**  uid : 0  gid : 0  **"noNewPrivileges":** true    **"root":**  "path": "rootfs", "readonly": true  **Docker daemon**  Cgroup Version: 2  **uid mapping**  0 0 1 | **Ressource Limits** "type": "RLIMIT\_NOFILE", "hard": 1024, "soft": 1024    **Resources**  **Devices** "allow": false, "access": "rwm"  **Masked Paths** : "/proc/acpi", "/proc/asound", "/proc/kcore", "/proc/keys", "/proc/latency\_stats", "/proc/timer\_list", "/proc/timer\_stats", "/proc/sched\_debug", "/sys/firmware", "/proc/scsi"  **Read-only Paths:** "/proc/bus", "/proc/fs", "/proc/irq", "/proc/sys", "/proc/sysrq-trigger"  N’est pas sécurisé , si une vulnérabilité affecte le démon , les conteneurs au-dessous sont affectés. |
| Crun (Le runtime de podman par défault) | **Namespaces**  pid , network , user, ipc , cgroup , mount , uts  **Capabilities**  Bounding : "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE"  Effective: "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE"  Inheritable:  Permitted : "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE"  Ambient: "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE"    **Capabilities** (sur un containeur podman exécuté )  Current: cap\_chown,cap\_dac\_override,cap\_fowner,cap\_fsetid,cap\_kill,cap\_setgid,cap\_setuid,cap\_setpcap,cap\_net\_bind\_service,cap\_sys\_  chroot,cap\_setfcap=ep  Bounding set =cap\_chown,cap\_dac\_override,cap\_fowner,cap\_fsetid,cap\_kill,cap\_setgid,cap\_setuid,cap\_setpcap,cap\_net\_bind\_service,cap  \_sys\_chroot,cap\_setfcap    **User**  uid : 0  gid : 0  **"noNewPrivileges":** true    **"root":**  "path": "rootfs", "readonly": true  **IDmappings**  gidmap:  - container\_id: 0  host\_id: 1000  size: 1  - container\_id: 1  host\_id: 524288  size: 65536  uidmap:  - container\_id: 0  host\_id: 1000  size: 1  - container\_id: 1  host\_id: 524288  size: 65536  cgroupVersion: v2 | **Ressource Limits** "type": "RLIMIT\_NOFILE", "hard": 1024, "soft": 1024    **Resources**  **Devices** "allow": false, "access": "rwm"  **Masked Paths** : "/proc/acpi", "/proc/asound", "/proc/kcore", "/proc/keys", "/proc/latency\_stats", "/proc/timer\_list", "/proc/timer\_stats", "/proc/sched\_debug", "/sys/firmware", "/proc/scsi"  **Read-only Paths:** "/proc/bus", "/proc/fs", "/proc/irq", "/proc/sys", "/proc/sysrq-trigger"  En mode rootless , il utilise fuse-overlay qui est une implémentation d’OverlayFS , elle est lente  CgroupsV1 ne peut pas être utilisé car il requiert des privilèges root.  MACVLAN, IPVLAN ne sont pas supportés en mode rootless. |
| RunC en mode rootless  Docker mode Rootless | **Namespaces**  pid , user , ipc , cgroup , mount , uts  **Capabilities** (runc spec)  Bounding : "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE"  Effective: "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE" Permitted : "CAP\_AUDIT\_WRITE", "CAP\_KILL", "CAP\_NET\_BIND\_SERVICE"  **Capabilities** (sur un containeur docker)  Current: cap\_chown,cap\_dac\_override,cap\_fowner,cap\_fsetid,cap\_kill,cap\_setgid,cap\_setuid,cap\_setpcap,cap\_net\_bind\_service,cap\_net\_  raw,cap\_sys\_chroot,cap\_mknod,cap\_audit\_write,cap\_setfcap=ep  Bounding set =cap\_chown,cap\_dac\_override,cap\_fowner,cap\_fsetid,cap\_kill,cap\_setgid,cap\_setuid,cap\_setpcap,cap\_net\_bind\_service,cap  \_net\_raw,cap\_sys\_chroot,cap\_mknod,cap\_audit\_write,cap\_setfcap  **User**  uid : 0  gid : 0  **"noNewPrivileges":** true    **"root":**  "path": "rootfs", "readonly": true  **Docker daemon**  Cgroup Version: 2  **IdMappings**  "uidMappings":  "containerID": 0, "hostID": 1000, "size": 1  "gidMappings":  "containerID": 0, "hostID": 1000, "size": 1 | **Ressource Limits** "type": "RLIMIT\_NOFILE", "hard": 1024, "soft": 1024    **Resources**  **Devices** "allow": false, "access": "rwm"  **Masked Paths** : "/proc/acpi", "/proc/asound", "/proc/kcore", "/proc/keys", "/proc/latency\_stats", "/proc/timer\_list", "/proc/timer\_stats", "/proc/sched\_debug", "/sys/firmware", "/proc/scsi"  **Read-only Paths:** "/proc/bus", "/proc/fs", "/proc/irq", "/proc/sys", "/proc/sysrq-trigger"  Les ports au dessous de 1024 ne  sont pas accessibles)  RunC utilise user namespace qui évite certains  Les tables d’IP variables , règles de firewall variable , interfaces réseaux  Il n’y a pas les capabilités comme  CAP\_SYS\_ADMIN  CAP\_DAC\_OVERRIDE  CAP\_NET\_OVERRIDE |
| udocker (en utilisant google collab)  Implémenté utilisant runc en mode rootless et crun e | **Capabilities (runc spec)**  bounding:  CAP\_AUDIT\_WRITE  CAP\_KILL  CAP\_NET\_BIND\_SERVICE  effective: "CAP\_AUDIT\_WRITE  CAP\_KILL  CAP\_NET\_BIND\_SERVICE  "permitted":  CAP\_AUDIT\_WRITE  CAP\_KILL  CAP\_NET\_BIND\_SERVICE"ambient": CAP\_AUDIT\_WRITE  CAP\_KILL  CAP\_NET\_BIND\_SERVICE  **Capabilities (sur un containeur udocker)**  Dépend sur les technologie de PRoot, Fakechroot, runc, crun et Singularity.  Est rootless par défaut. | **Ressource Limits** "type": "RLIMIT\_NOFILE", "hard": 1024, "soft": 1024  **Resources**  **Devices** "allow": false, "access": "rwm"  **Masked Paths** : "/proc/acpi", "/proc/asound", "/proc/kcore", "/proc/keys", "/proc/latency\_stats", "/proc/timer\_list", "/proc/timer\_stats", "/proc/sched\_debug", "/sys/firmware", "/proc/scsi"  **Read-only Paths:** "/proc/bus", "/proc/fs", "/proc/irq", "/proc/sys", "/proc/sysrq-trigger"  Même limitations d’une technologie rootless car il dépend sur crun.    Car il ne dépend pas sur les namespaces niveau kernel comme  DOcker, il n’offre pas une isolation complète  ou un niveau d’intégration complet |
|  |  |  |