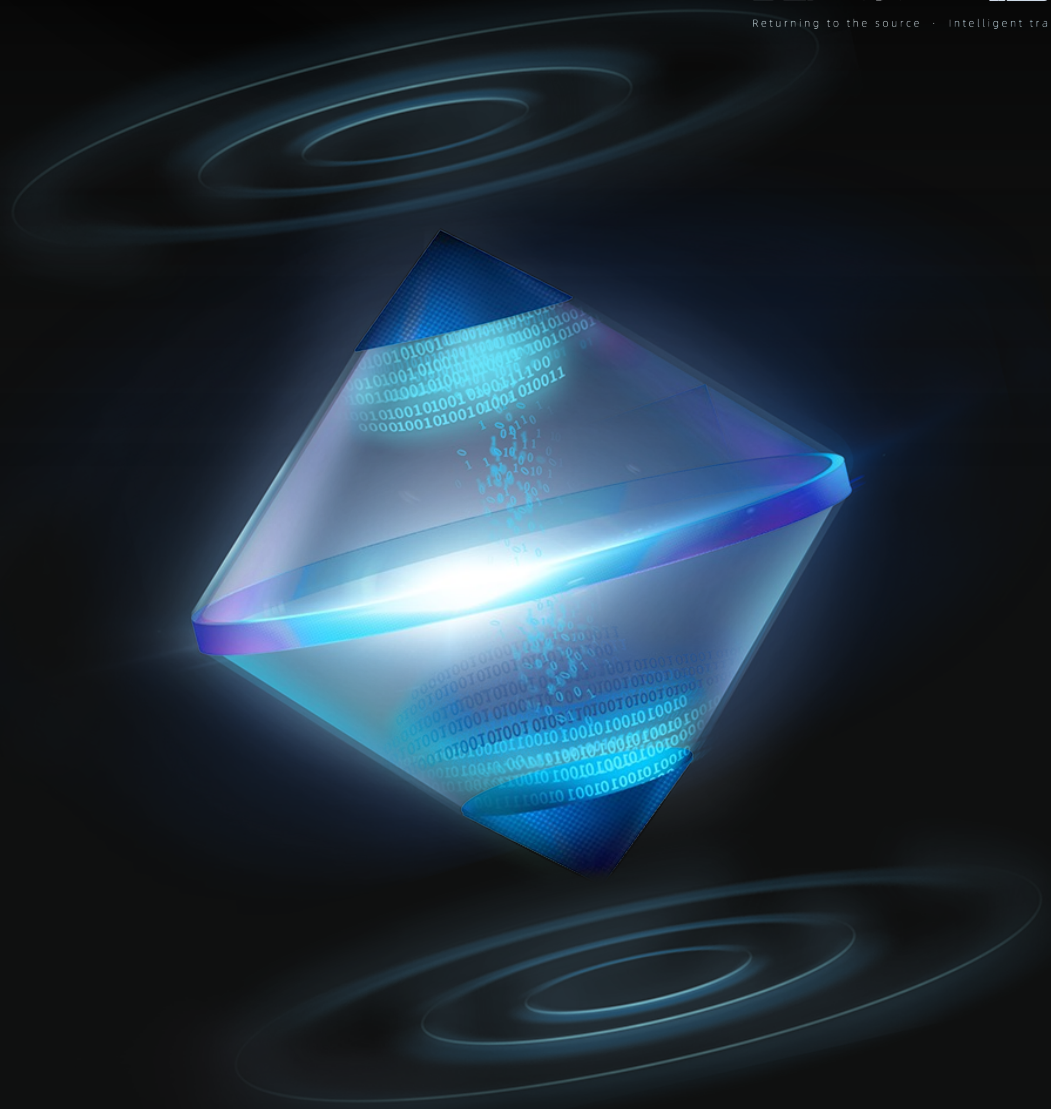


# Modern Linux sandboxing technology

李强, 杨玉彪  
2023/08/20



## 关于我们

- 蚂蚁集团安全专家/研发专家
- 安全研究与安全建设
- Linux内核/虚拟化/容器/云原生
- KCon, HITB, CanSecWest...



## 目录

- 安全沙箱简介
- Linux沙箱机制与方案
- gVisor简介
- 基于gVisor构建安全沙箱
- 未来规划

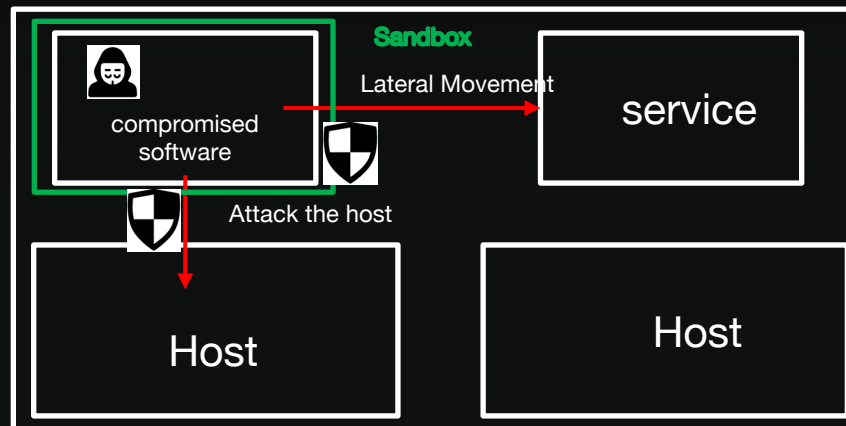
# 01 | 安全沙箱简介

## 什么是安全沙箱

- 一种安全机制/方案，能够隔离的运行程序
- 经常用来限制资源或者运行不受信任的三方程序
- 各个操作系统上都有多种实现
- 沙箱的含义很多，这里主要讨论进程级安全沙箱
- 沙箱是安全领域一个非常古老的话题，一直在不断进化

## 沙箱限制什么？

- 进程
- 文件系统
- 网络访问
- 能力(Capabilities)
- CPU/Memory/IO/Devices



## 沙箱的使用场景

- 外部用户可控代码
- 不信任的第三程序
- 解析类的程序：经常会有漏洞
- 恶意软件分析

## 沙箱的需求

- 沙箱的使用场合
  - 物理机
  - 虚拟机
  - 容器
- 需要能够拦截横向移动
  - 网络策略
- 需要能够阻止纵向移动
  - 内核隔离
  - 系统安全策略



## 02 | Linux沙箱机制与方案

## 机制：setuid

- setuid是文件的一个标志位
- 当可执行文件设置该位时，启动的进程会有文件owner的权限
- setuid通常的使用场景：非特权用户需要进行特权操作
- 沙箱通常都是setuid程序，因为需要为运行的程序设置沙箱环境
- BTW: setuid root程序很多时候容器有漏洞，比如pwnkit

## 机制：ptrace

- ptrace是Linux中的一个系统调用
- 进程可以通过ptrace控制其他进程
- ptrace可以修改进程的内存以及控制流
- ptrace通常用来实现调试器，比如gdb
- 沙箱可以使用ptrace来控制需要被沙箱的程序
- BTW: strace依赖于ptrace

## 机制：seccomp

- seccomp是Linux中的有名的安全机制
- seccomp用来限制进程能够触发的系统调用
- 内核的大部分功能都是通过syscall暴露到用户态
- 大多数的进程只需要使用很少的一部分syscall
- seccomp可以用来对不需要使用的syscall进行限制
- 沙箱可以使用seccomp来限制被沙箱进程的syscall
- BTW: seccomp在很多软件中都有使用，比如QEMU

## 机制：capabilities

- capabilities是Linux中的安全机制，用来将权限进行划分
- 传统的权限检查为root或者非root, root具有所有权限
- capabilities提供了进程对内核资源的细粒度管控
- 典型cap: CAP\_SYS\_ADMIN, CAP\_SYS\_MODULE, CAP\_NET\_ADMIN
- 沙箱通常都需要限制被沙箱进程的capabilities
- BTW: Capabilities在容器生态中广泛使用

## 机制：chroot

- chroot是Linux中的一个系统调用
- chroot用来修改进程的root目录
- chroot之后的进程只能看到新的root下的目录以及文件，外部的无法看到
- 沙箱经常使用chroot来为被沙箱进程提供一个单独的文件系统视角
- BTW: chroot在容器生态中广泛使用

## 机制：namespace

- namespace是Linux中的一个机制
- 不同namespace中的进程看到不同的系统资源
- Linux有很多类型的namespace: PID, NET, MOUNT, UTS, USER, IPC等
- 沙箱经常使用namespace来对进程进行隔离
- BTW: namespace是容器的基础技术之一

## 机制：cgroup

- cgroup是Linux中的一个机制
- cgroup用来限制进程能够使用的系统资源
- cgroup种类：CPU, Memory, Disk IO, Network, Devices等
- 沙箱经常使用cgroup来限制被沙箱进程的资源使用
- BTW: cgroup是容器的基础技术之一



## 机制：netfilter

- netfilter是内核子系统
- netfilter用于网络数据包的过滤与修改
- netfilter提供了多个hook点允许程序注册
- 网络数据包在协议栈上处理时，会依次调用相关hook函数
- 沙箱通常使用netfilter/iptables来做网络隔离

## 机制：MAC

- Mandatory Access Control, 强制访问控制系统
- Linux中的MAC都是基于Linux Security Module(LSM)
- Linux中有多个实现：SELinux, Smack, AppArmor
- 进程访问内核资源时，MAC实现的LSM hook点函数会被调用
- MAC会决定该访问是否允许
- 使用MAC做沙箱需要熟悉非常复杂的MAC策略语言

## 方案：setuid-sandbox

- 允许被沙箱进程放弃部分权限
- 基于namespace的UID隔离
- chroot
- 参考：<https://code.google.com/archive/p/setuid-sandbox/>

```
test@ubuntu:~/setuid-sandbox$ ./sandboxme -u4 /bin/sh
Helper: write to 4 ($SBX_D) to chroot the sandboxed process
Could not find user suidsandbox
Hi from the sandbox! I'm pid=1, uid=1000, gid=1000, dumpable=Y
Executing /bin/sh
$ echo C>&$SBX_D
$ Helper: I chrooted you
ls /
sh: 2: ls: not found
```

## 方案：systemd

- systemd为服务提供了多种沙箱选择
- 用于限制沙箱的资源访问
- ProtectSystem=yes: /usr、/boot 只读
- ProtectDevics=yes: /dev namespace开启
- ReadOnlyDirectories= : 指定文件只读
- PrivateNetwork=yes: 没有外部网络访问权限
- systemd 使用 namespace/seccomp, BPF-LSM

```
[Service]
ProtectSystem=strict
ProtectHome=yes
PrivateDevices=yes
ProtectKernelTunables=yes
ProtectKernelModules=yes
ProtectControlGroups=yes
SystemCallFilter=@system-service
SystemCallErrorNumber=EPERM
NoNewPrivileges=yes
PrivateTmp=yes
```

## 方案：nsjail

- 非常有名的轻量级的进程隔离工具（沙箱）
- 典型的使用Linux namespace以及seccomp-bpf
- 为进程提供namespace/filesystem等资源的隔离
- 提供网络服务与本地进程之间的隔离
- 与宿主机共享内核
- 没有细粒度的网络访问策略

## 方案：firejail

- 跟nsjail非常类似
- 为不受信任的程序提供一个受限的运行环境
- 同样使用namespaces, seccomp-bpf以及Linux capabilities等
- 可以运行任意类型的进程：服务进程、GUI进程等
- 与宿主机共享内核
- 没有细粒度的网络访问策略

## 机制和方案都很多

- 但是他们都是基于共享内核的
- 几乎所有方案都缺乏网络策略

## 我们需要什么样的沙箱？

- 进程限制：限制能够执行的程序
- 文件系统访问限制：限制文件的读写权限
- 网络访问权限：细粒度的网络管控策略（ip/端口/域名）
- 内核隔离：需要使用独立内核

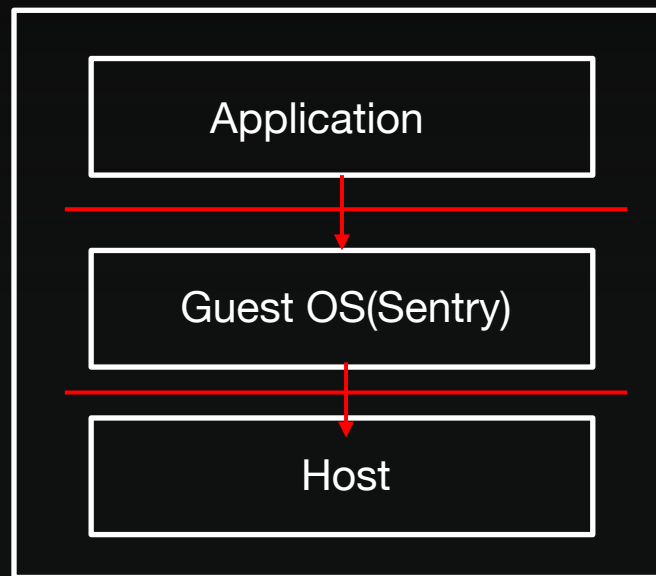
总结：我们需要横向和纵向的强隔离



## 03 | gVisor简介

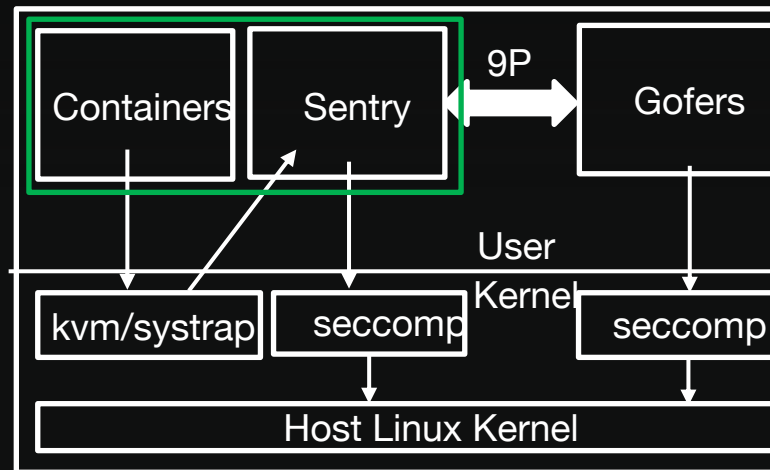
## gVisor总览

- gVisor是一个应用内核
- 使用Go实现, 内存安全
- 实现了大部分的Linux syscall接口, Sentry
- 大部分的Linux应用都能跑
- 实现了OCI spec



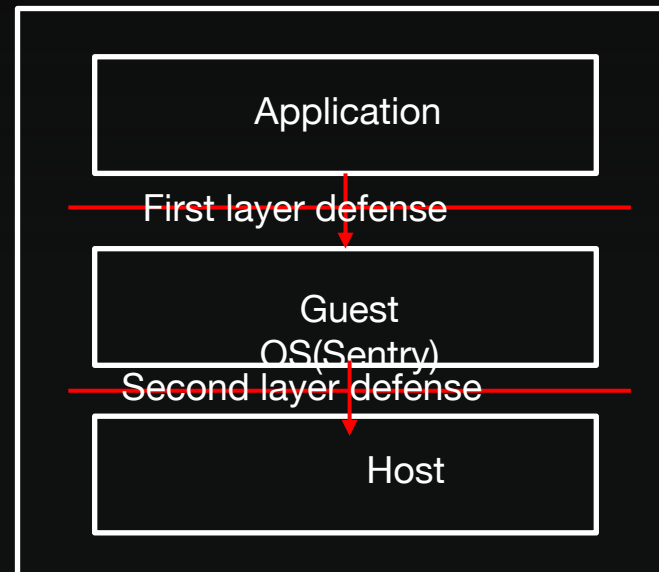
## gVisor纵深防御

- Sentry: 虚拟机内核, 第一层防御
- Sentry使用ptrace/KVM/systrap拦截syscall
- Gofers: 虚拟机与宿主机文件系统共享
- Sentry/Gofers: 都有多种基本安全原语保护
- seccomp/capabilities/chroot/namespace/cgroup, 第二次防御



## gVisor纵深防御

- 第一层防御
  - Sentry: 虚拟机内核, 处理容器中大部分的系统调用
  - 内存安全: 没有缓冲区溢出, 没有UAF
  - Gofers: 虚拟机与宿主机文件系统共享
- 第二层防御
  - Sentry/Gofers到宿主机的访问有多重安全机制
    - seccomp
    - namespace
    - cgroup



## gVisor作为安全沙箱缺啥

- gVisor主要使用在云原生/容器生态
- gVisor实现了OCI spec
- OCI spec中有部分容器安全相关的配置，但是作为安全沙箱还缺乏安全能力
- OCI spec没有网络相关的配置，这部分是由CNI的network policy指定的
- 总结：gVisor只是有了纵向的隔离，解决容器逃逸等问题，但是缺乏横向隔离

## gVisor定制化

- gVisor是使用Go写的应用内核
- 可以非常方便的进行定制化需求
- 例子：阻止/usr/bin/ls的执行

```

root@test-VirtualBox:/home/test/test11# ./runsc --debug run abc
/usr/bin/ls
execute: /usr/bin/ls
deny ls
sh: 1: /usr/bin/ls: Operation not permitted

```

```

root@test-VirtualBox:/home/test/gvisor# git diff
diff --git a/pkg/sentry/syscalls/linux/sys_thread.go b/pkg/sentry/
/sys_thread.go
index 9b448821f..0c1bb53b0 100644
--- a/pkg/sentry/syscalls/linux/sys_thread.go
+++ b/pkg/sentry/syscalls/linux/sys_thread.go
@@ -15,6 +15,7 @@
package linux

import (
+   "fmt"
   "gvisor.dev/gvisor/pkg/abi/linux"
   "gvisor.dev/gvisor/pkg/errors/linuxerr"
   "gvisor.dev/gvisor/pkg/fspath"
@@ -149,6 +150,11 @@ func execveat(t *kernel.Task, dirfd int32, pa
gvAddr, envvAddr host
           pathname = executable.MappedName(t)
       }
+
+   fmt.Println("execute: ", pathname)
+   if pathname == "/usr/bin/ls" {
+       fmt.Println("deny ls")
+       return 0, nil, linuxerr.EPERM
+   }
// Load the new TaskImage.
wd := t.FSContext().WorkingDirectory()
defer wd.DecRef(t)

```

## 04 | 基于gVisor构建安全沙箱

## 需求


- 构建一个具备横向和纵向防御能力的安全沙箱
- 传统方案两个方面都有所不足
- gVisor实现了纵向的安全防御
- 但是gVisor本身没有实现网络管控，缺乏横向防御
- gVisor定制能力强，可以自己定制



## 想法是否已经被实现?

- Firejail的一个issue提到了我们的想法
- 但是并没有实现

### [Feature request] gVisor backend #3942

 Open ghost opened this issue on Feb 2, 2021 · 3 comments



ghost commented on Feb 2, 2021 · edited by ghost ▾

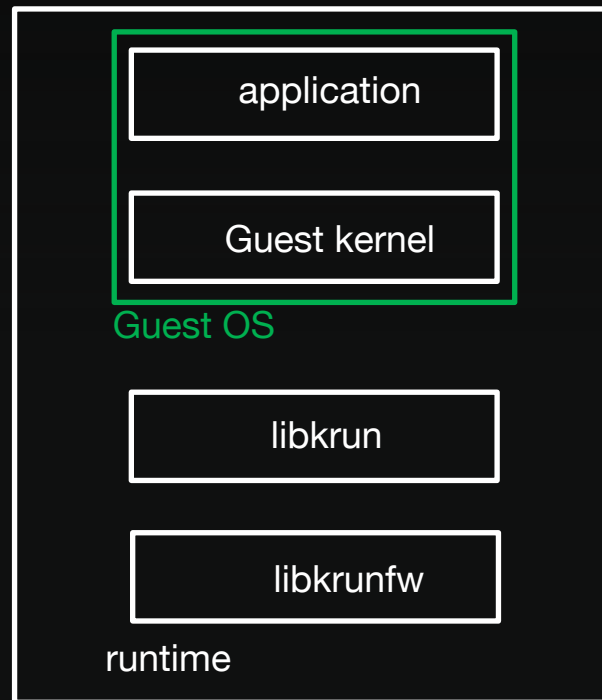
gVisor emulates the majority of linux syscalls in userland, providing a respectable sandbox.

gVisor provides a runtime (runsc) capable of running OCI spec containers. [https://gvisor.dev/docs/user\\_guide/quick\\_start/oci/](https://gvisor.dev/docs/user_guide/quick_start/oci/)

It should be possible to either modify gVisor to accept a different interface or to have firejail output an OCI config for an OCI runtime.

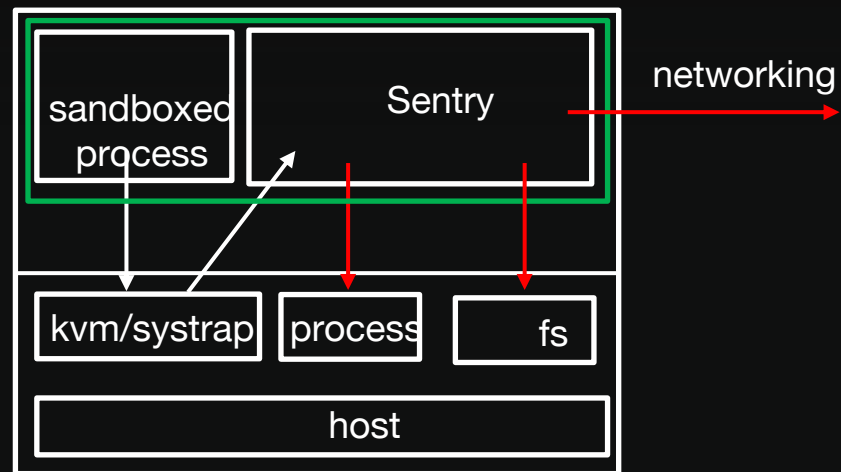
## 想法是否已经被实现?

- libkrun:动态的连接库
- 提供隔离能力, 能将程序跑到虚拟机中
- 看起来跟gVisor类似, 增加了纵向的隔离
- 但是缺乏横向隔离



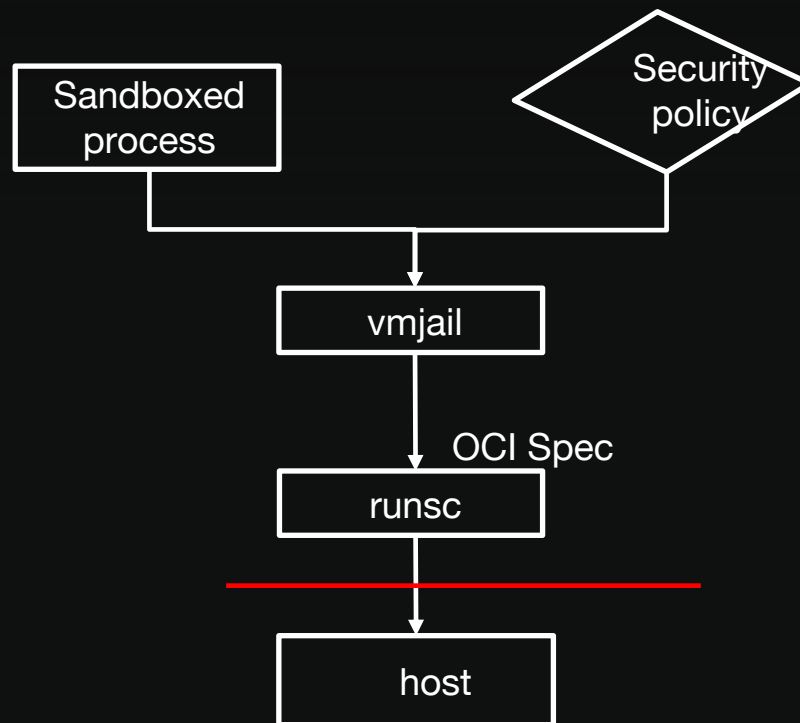
## vmjail 简介

- vmjail 是一款基于gVisor的进程级安全沙箱
- 是一个setuid程序，用来创建沙箱环境
- 横向隔离
  - 基于gVisor
  - 自定义了网络策略
- 纵向隔离
  - 基于gVisor
  - 定义文件/进程的策略



## vmjail 架构

- vmjail 安全策略
  - Process/file/networking
  - Memory/CPU
- vmjail 策略 -> OCI spec
- runc: 启动Sentry和Gofer
- Sentry: 安全策略的检测
  - OCI spec
  - 定制化



## OCI 介绍

- Open Container Initiative: 包括多种spec, 容器运行时、镜像
- 主要定义了一个容器应该怎么被运行起来
- OCI有多种实现, 最著名的runc
- OCI实现是云原生生态的最底层的组件
- OCI spec中有部分安全相关的策略
- vmjail可以直接使用这部分安全策略

## vmjail 策略 -> OCI spec

- vmjail 策略包含所有沙箱安全策略：进程/文件/Memory/CPU等
- 部分策略直接转换为OCI spec
- 其他策略在Sentry中实现

```

{
  "destination": "/tmp",
  "type": "bind",
  "source": "/tmp",
  "options": [
    "rbind",
    "rw"
  ]
}

```

```

"security": {
  "network": {
    "mode": "hostwithpolicy",
    "policy": {
      "listen": [80],
      "tcp": ["1.1.1.1:"],
      "udp": ["*:53"],
      "dns": ["npm.org:*", "python.org:*"]
    }
  },
  "file": {
    "writablePaths": ["/tmp"],
    "maskedPaths": ["/mnt"]
  },
  "process": {
    "allow": ["/usr/bin/ls"]
  }
}

```

```

"namespaces": [
  {
    "type": "pid"
  },
  {
    "type": "ipc"
  },
  {
    "type": "uts"
  },
  {
    "type": "mount"
  }
],
"maskedPaths": [
  "/mnt"
]
}

```

## 文件系统限制

- 定义了被沙箱进程的能够对文件系统的访问权限
- vmjail实现了如下策略：
  - rootfs只读：大部分的目录不能被写入
  - 提供writeablePaths: 定义可以写入的文件和目录
  - 提供maskedPaths: 定义被沙箱进程不可见的文件和目录

## 文件系统限制-OCI实现

- OCI满足vmjail的需求
- rootfs只读可以通过配置文件中：`.root.readonly`配置为true实现
- writeablePaths可以通过配置mounts实现
- maskedPaths可以通过配置文件：`.linux.maskedPaths`实现

```
    "root": {  
      "path": "rootfs",  
      "readonly": true  
    },  
  },
```

```
    "mounts": [  
      {  
        "destination": "/tmp",  
        "type": "bind",  
        "source": "/tmp",  
        "options": [  
          "rbind",  
          "rw"  
        ]  
      }  
    ],
```

```
    "linux": {  
      "maskedPaths": [  
        "/mnt"  
      ]  
    }  
  }  
}
```



## 文件系统限制-vmjail

- vmjail 使用OCI spec中的文件配置
- 相关策略从vmjail 策略转换为OCI spec

```

"system": {
  "runtime": "runsc",
  "rollback": "",
  "accel": "ptrace",
  "root": {
    "path": "/",
    "readonly": true
  },

```

```

"security": {
  "network": {
    "mode": "hostwithpolicy",
    "policy": {
      "listen": [80],
      "tcp": ["1.1.1.1:"],
      "udp": ["*:53"],
      "dns": ["npm.org:*", "python.org:*"]
    }
  },
  "file": {
    "writablePaths": ["/tmp"],
    "maskedPaths": ["/mnt"]
  },
  "process": {
    "allow": ["/usr/bin/ls"]
  }
}

```

## 网络限制

- 定义被沙箱进程能够进行的网络访问
- vmjail 实现了如下策略：
  - 完全没有网络访问
  - 限制出向的IP/端口
  - 限制出向的域名
  - 限制监听的端口

```
"security": {  
  "network": {  
    "mode": "none",  
    "policy": {  
      "listen": [80],  
      "tcp": ["1.1.1.1:"],  
      "udp": ["*:53"],  
      "dns": ["*.npm.org:*", "*.python.org:*"]  
    }  
  },  
}
```

## 网络限制-CNI

- OCI 对网络策略没有规定
- Container Network Interface(CNI)定义了网络策略
- CNI的network policy定义了pod/容器之间的网络策略控制
- 进程级沙箱中使用CNI太重

## 网络限制-vmjail

- 使用gVisor的host 网络栈 (--network host)
- 修改gVisor代码
- 当gVisor运行时，通过命令行传递网络策略文件
- 当应用触发网络行为时，检查是否符合策略



## 进程限制

- 大多数的被沙箱进程都只运行一个程序
- 没有反弹shell，没有其他工具能够运行
- 该功能大部分沙箱都没有，vmjail也处于开发中
- 可执行文件的全路径作为策略

## 进程限制-OCI

- OCI没有对进程执行进行限制
- 可以通过OCI spec中的maskedPaths去实现
- 但是该方法是黑名单，我们需要白名单模式

## 进程限制-vmjail

- vmajil 策略定义了哪些可执行文件能够执行
- 修改gVisor代码
- 当运行gVisor时，通过命令行传递策略
- 当应用程序执行不在白名单策略中的文件时，拒绝执行

## CPU/Memory/Devices

- OCI spec 定义了这些资源的限制
- vmjail 可以直接使用
- CPU/Memory/Devices/Capabilities

```

"system": {
  "runtime": "runc",
  "rollback": "",
  "accel": "ptrace",
  "root": {
    "path": "/",
    "readonly": true
  },
  "memory": {
    "max": 1073741824
  },
  "cpu": {
    "number": 4
  }
},

```

```

],
"resources": {
  "cpu": {
    "period": 100000,
    "quota": 400000
  },
  "memory": {
    "limit": 1073741824
  }
}
},

```



## 实现中的一些问题

- gVisor的问题
  - wget can't connect to https websites in host network mode #8156
  - statx syscall is not supported before Linux 4.11 #8229
  - gVisor cgroup的删除等待5s
  - gVisor没有实现readOnlyPaths和maskedPaths
- 按照当前用户权限执行
  - getuid, 传递给OCI spec
- gVisor要求4.14内核运行
  - vmjail 允许在不支持的内核上运行时回滚

## 示例

- 独立的内核
- rootfs只读

```
'system': {  
  "runtime": "runsc",  
  "rollback": "",  
  "accel": "ptrace",  
  "root": {  
    "path": "/",  
    "readonly": true  
  },  
  "..."  
}
```

```
root@test-VirtualBox:/home/test/test# ./vmjail -c security.json dmesg  
[ 0.000000] Starting gVisor...  
[ 0.410024] Mounting deweydecimalfs...  
[ 0.765850] Daemonizing children...  
[ 1.124008] Creating cloned children...  
[ 1.585270] Preparing for the zombie uprising...  
[ 1.742580] Segmenting fault lines...  
[ 1.753824] Forking spaghetti code...  
[ 1.782736] Rewriting operating system in Javascript...  
[ 2.097989] Constructing home...  
[ 2.544215] Moving files to filing cabinet...  
[ 2.807455] Waiting for children...  
[ 3.269826] Setting up VFS2...  
[ 3.739661] Ready!  
root@test-VirtualBox:/home/test/test# ./vmjail -c security.json -- touch /abc  
touch: cannot touch '/abc': Read-only file system  
root@test-VirtualBox:/home/test/test# ./vmjail -c security.json -- touch /home/abc  
touch: cannot touch '/home/abc': Read-only file system
```

## 示例

- writablePaths
- maskedPaths

```
},
"file": {
  "writablePaths": ["/tmp"],
  "maskedPaths": ["/var"]
},
```

```
root@test-VirtualBox:/home/test/src/test# ./vmjail -c security.json touch /abc
touch: cannot touch '/abc': Read-only file system
root@test-VirtualBox:/home/test/src/test# ./vmjail -c security.json touch /tmp/abc
root@test-VirtualBox:/home/test/src/test# ./vmjail -c security.json echo aaa >> /tmp/abc
root@test-VirtualBox:/home/test/src/test# cat /tmp/abc
aaa
root@test-VirtualBox:/home/test/src/test# ls -lh /var
total 48K
drwxr-xr-x  2 root root    4.0K  8月 17 08:32 backups
drwxr-xr-x 16 root root    4.0K  7月 22 19:22 cache
drwxrwsrwt  2 root whoopsie 4.0K  8月 16 13:38 crash
drwxr-xr-x 71 root root    4.0K  8月 17 14:42 lib
drwxrwsr-x  2 root staff   4.0K  4月 18  2022 local
lrwxrwxrwx  1 root root      9  7月 22 13:03 lock -> /run/lock
drwxrwxr-x 13 root syslog  4.0K  8月 14 19:15 log
drwxrwsr-x  2 root mail    4.0K  4月 19  2022 mail
drwxrwsrwt  2 root whoopsie 4.0K  4月 19  2022 metrics
drwxr-xr-x  2 root root    4.0K  4月 19  2022 opt
lrwxrwxrwx  1 root root      4  7月 22 13:03 run -> /run
drwxr-xr-x 11 root root    4.0K  7月 23 12:49 snap
drwxr-xr-x  6 root root    4.0K  7月 22 13:05 spool
drwxrwsrwt 11 root root    4.0K  8月 17 15:25 tmp
root@test-VirtualBox:/home/test/src/test# ./vmjail -c security.json ls /var
/var
```

## 示例-网络

- 不能访问白名单以外的域名

```
"security": {
  "network": {
    "mode": "hostwithpolicy",
    "policy": {
      "listen": [80],
      "tcp": ["1.1.1.1:"],
      "udp": ["*:53"],
      "dns": ["*.npm.org:*", "*.python.org:*"]
    }
  }
},
```

```
root@test-VirtualBox:/home/test/test# wget pypi.org
--2023-08-04 10:36:46-- http://pypi.org/
Resolving pypi.org (pypi.org)... 151.101.64.223, 151.101.0.223, 151.101.192.223, ...
Connecting to pypi.org (pypi.org)|151.101.64.223|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://pypi.org/ [following]
--2023-08-04 10:36:46-- https://pypi.org/
Connecting to pypi.org (pypi.org)|151.101.64.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23162 (23K) [text/html]
Saving to: 'index.html.1'

index.html.1          100%[=====>] 22.62K  --.-KB/s

2023-08-04 10:36:46 (2.61 MB/s) - 'index.html.1' saved [23162/23162]

root@test-VirtualBox:/home/test/test# ./vmjail -c security.json wget pypi.org
URL transformed to HTTPS due to an HSTS policy
--2023-08-04 10:36:54-- https://pypi.org/
Resolving pypi.org (pypi.org)... 2a04:4e42:600::223, 2a04:4e42:200::223, 2a04:4e42:400::223, ...
Connecting to pypi.org (pypi.org)|2a04:4e42:600::223|:443... failed: Operation not permitted.
Connecting to pypi.org (pypi.org)|2a04:4e42:200::223|:443... failed: Operation not permitted.
Connecting to pypi.org (pypi.org)|2a04:4e42:400::223|:443... failed: Operation not permitted.
Connecting to pypi.org (pypi.org)|2a04:4e42::223|:443... failed: Operation not permitted.
Connecting to pypi.org (pypi.org)|151.101.128.223|:443... failed: Operation not permitted.
Connecting to pypi.org (pypi.org)|151.101.192.223|:443... failed: Operation not permitted.
Connecting to pypi.org (pypi.org)|151.101.0.223|:443... failed: Operation not permitted.
Connecting to pypi.org (pypi.org)|151.101.64.223|:443... failed: Operation not permitted.
```

## 示例-网络

- 能够访问白名单照顾中的域名

```
"security": {
  "network": {
    "mode": "hostwithpolicy",
    "policy": {
      "listen": [80],
      "tcp": ["1.1.1.1:"],
      "udp": ["*:53"],
      "dns": ["*.npm.org:*", "*.python.org:*"]
    }
  },
},
```

```
^Croot@test-VirtualBox:/home/test/test# ./vmjail -c security.json wget npm.org
--2023-08-04 10:37:17-- http://npm.org/
Resolving npm.org (npm.org)... 72.167.71.164
Connecting to npm.org (npm.org)|72.167.71.164|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://npm.org/ [following]
--2023-08-04 10:37:18-- https://npm.org/
Connecting to npm.org (npm.org)|72.167.71.164|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
index.html.2: Read-only file system

Cannot write to 'index.html.2' (Success).
root@test-VirtualBox:/home/test/test# ./vmjail -c security.json wget python.org
--2023-08-04 10:37:26-- http://python.org/
Resolving python.org (python.org)... 151.101.0.223, 151.101.128.223, 151.101.64.223, ...
Connecting to python.org (python.org)|151.101.0.223|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.python.org/ [following]
--2023-08-04 10:37:26-- https://www.python.org/
Resolving www.python.org (www.python.org)... 151.101.76.223, 2a04:4e42:12::223
Connecting to www.python.org (www.python.org)|151.101.76.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 50260 (49K) [text/html]
index.html.2: Read-only file system

Cannot write to 'index.html.2' (Success).
```

## 示例-网络

- 没有网络

```
},  
"security": {  
  "network": {  
    "mode": "none",  
    "policy": {  
      "listen": [80],  
      "tcp": ["1.1.1.1:"],  
      "udp": ["*:53"],  
      "dns": ["*.npm.org:*", "*.python.org:*"]  
    }  
  }  
},
```

```
root@test-VirtualBox:/home/test/test# ./vmjail -c security.json wget python.org  
--2023-08-04 10:42:17-- http://python.org/  
Resolving python.org (python.org)... failed: Temporary failure in name resolution.  
wget: unable to resolve host address 'python.org'  
root@test-VirtualBox:/home/test/test# ./vmjail -c security.json ip a  
root@test-VirtualBox:/home/test/test#
```

# 05 | 未来规划

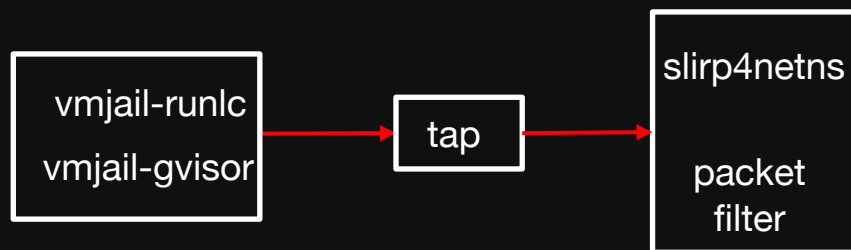
## 更多的runtime支持

- 基于gVisor的安全沙箱从安全角度来讲是完美的
- 当时世界并不是总是关于安全的
- vmjail 在性能关键的场合受影响比较大
- 部分任务更关心性能而不是安全
- vmjail 需要舍弃部分安全，以增加性能
- runlc, 轻量级的沙箱runtime, 基于namespace/cgroup等传统技术



## 统一的网络管控

- vmjail 增加runlc后端作为增强性能的一个选择
- 但是传统方案没有网络策略
- vmjail 准备使用用户态网络栈: slirp, past
- 在用户态网络栈做网络策略
- 类似CNI, 但是在更底层



## gVisor用于进程的动态分析

- 动态分析是沙箱的另一个场景
- gVisor可以监视进程发生的所有行为
- 进程/网络/文件行为
- 可以使用gVisor做恶意软件分析

```
root@test-VirtualBox:/home/test/test11# ./runsc --pod-init-config=./gvisor/examples/seccheck/pod_init.json run abc
ls
bin
boot
cdrom
```

```
root@test-VirtualBox:/home/test/gvisor# ./bazel-bin/examples/seccheck/server_cc
Socket address /tmp/gvisor_events.sock
Connection accepted
Start => id: "abc" cwd: "/" args: "sh"
E Open sysno: 257 fd: -100 pathname: "/etc/ld.so.cache" flags: 524288
X Open exit { result: 3 } sysno: 257 fd: -100 pathname: "/etc/ld.so.cache" flags: 524288
E Open sysno: 257 fd: -100 pathname: "/lib/x86_64-linux-gnu/libc.so.6" flags: 524288
X Open exit { result: 3 } sysno: 257 fd: -100 pathname: "/lib/x86_64-linux-gnu/libc.so.6" flags: 524288
E Read context_data { time_ns: 1690967929600693083 thread_id: 1 container_id: "abc" }
X Read exit { result: 832 } fd: 3 count: 832
E Read context_data { time_ns: 1690967929610689075 thread_id: 1 container_id: "abc" }
X Read exit { result: 3 } count: 8192
CloneInfo => created_thread_id: 2 created_thread_group_id: 2 created_thread_start_time_ns: 1690967929610689075
E Open sysno: 257 fd: -100 pathname: "/etc/ld.so.cache" flags: 524288
X Open exit { result: 3 } sysno: 257 fd: -100 pathname: "/etc/ld.so.cache" flags: 524288
E Open sysno: 257 fd: -100 pathname: "/lib/x86_64-linux-gnu/libselinux.so.1" flags: 524288
X Open exit { result: 3 } sysno: 257 fd: -100 pathname: "/lib/x86_64-linux-gnu/libselinux.so.1" flags: 524288
```

## 最终规划

- vmjail将会有两种模式
- 用来运行不受信任代码
  - 基于gVisor的runtime: gVisor, 专注于安全
  - 基于namespace/cgroup等传统的runtime: runlc, 专注于性能
  - 每种方案都会有横向和纵向的安全隔离
- 用来进行程序的动态分析

## 总结

- 当前的沙箱方案缺乏部分安全相关的特性
- gVisor是一个完整的沙箱方案
- gVisor自身缺乏部分安全策略
- gVisor可以轻松进行定制用来满足各种安全需求
- 基于gVisor可以构建具有横向和纵向的强安全隔离的进程级安全沙箱

# 感谢您的观看！

THANK YOU FOR YOUR WATCHING

