gVisor: Modern Linux Sandboxing Technology

Li Qiang | Senior Security Engineer, AntGroup

24 Aug 2023

About me

- Platform infrastructure security engineer @AntGroup
- Security researcher && Developer
- All low-level materials: kernel/virtualization/container/security
- Speaker of HITB, Ruxcon, Syscan360

Agenda

- Introduction to sandbox
- Linux sandbox mechanisms and solutions
- gVisor overview
- Build sandbox based gVisor
- The future

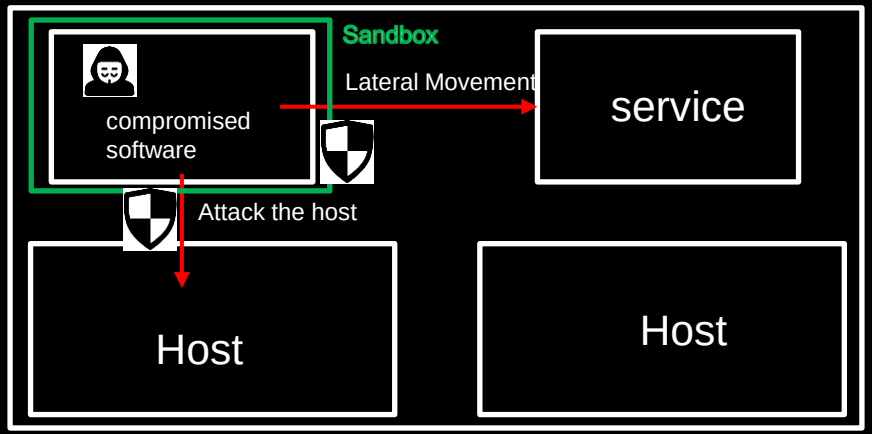
01 | Introduction to sandbox

Sandbox-what it is

- A security mechanism for separating running programs
- Mostly used to restrict system resources which untrusted program can access
- A lot of implementation and use cases
- This talk about Linux application sandboxing-process level sandbox
- Sandbox is a very old topic in security area

Sandbox-what it restricts

- Process
- File system
- Network access
- Capabilities
- CPU/Memory/IO/Devices



Sandbox-use cases

- Attacker controlled code
- Untrust third party program
- Vulnerable parser: it often has been found vulnerabilities
- Malware analysis

Sandbox-realworld needs

- Should be used in a lot of place
  - Bare mental machine
  - Virtual Machine
  - Container
- Should defense against lateral movement
  - Network security policy
- Should defense against vertical escape
  - Kernel isolation
  - System security policy

02 | Linux sandbox mechanisms and solutions

Mechanism: setuid

- File flag about a file
- When set on exec file, the process will have the file owner’s privileges
- Mostly used to do some privileged task by unprivileged user
- Sandbox uses this often because it needs setup sandbox environment
- BTW: setuid root program vuln often leads privilege escalation such as pwnkit

Mechanism: ptrace

- ptrace is a linux syscall
- One process can use ptrace to control another process
- ptrace can change process’s memory and control flow
- Mostly used to implement debugger such as gdb
- Sandbox can use ptrace to total control the sandboxed program
- BTW: the famous strace uses ptrace

Mechanism: seccomp

- seccomp is a Linux security facility
- seccomp can be used to restrict the syscall the process can trigger
- The kernel has a lot of function which exposed by syscall
- Most process uses only part of the syscall
- Seccomp can be used to reduce attack surface by limiting the syscall
- Sandbox can use seccomp to restrict the sandboxed process’s syscall
- BTW: seccomp is used in a lot of software such as QEMU

Mechanism: capabilities

- Capabilities is a Linux mechanism which divides privileges into units
- Traditional permission check gives the root user all permissions
- Capabilities allow process have fine-grained access to kernel resources
- CAP_SYS_ADMIN, CAP_SYS_MODULE, CAP_NET_ADMIN and so on
- Sandbox often needs to restrict the sandboxed process’s capabilities
- BTW: capabilities is used in container ecosystem heavily

Mechanism: chroot

- chroot is a Linux syscall
- chroot changes the caller process’s root directory
- The chrooted process can only see the file system begin with the new root
- Sandbox often needs to provide an isolated filesystem view to sandboxed process
- BTW: chroot is used in container ecosystem heavily

Mechanism: namespaces

- Namespaces is a Linux mechanism
- Process in different namespaces sees different kernel resources
- PID, NET, MOUNT, UTS, USER, IPC and so on
- Sandbox often uses namespaces to isolate different process
- BTW: Namespaces are a fundamental tech of containers

Mechanisms: cgroup

- cgroup is a Linux mechanism
- Which restrict the system resource that process can consume
- CPU, Memory, Disk IO, Network, Devices and so on
- Sandbox often uses this to limit sandboxed process’s system resource usage
- BTW: cgroup are a fundamental tech of containers

Mechanisms: Netfilter

- Netfilter is a kernel subsystem
- Netfilter is used to packet filtering and mangling
- Netfilter provides hook points which allow programs to register
- As Packets go through the stack, every registered hook will get a chance to process it
- Sandbox often uses netfilter/iptables to do network isolation

Mechanisms: MAC

- Mandatory Access Control
- MAC is based Linux Security Module(LSM) in linux
- Several implementation: SELinux, Smack, AppArmor
- When the process access the kernel resource, security hook in MAC will be called
- Then do the pass/reject decision according to predefined security policy
- The security policy is quite complicated

Solution: setuid-sandbox

- A sandbox allow the sandboxed program to drop privileges
- UID isolation(namespace)
- Chroot
- More info: 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
```

Solution: systemd

- systemd also provide a lot of sandbox options for services
- So the service process has a limited access to system resource
- ProtectSystem=yes: /usr、/boot read-only
- ProtectDevics=yes: private /dev namespace
- ReadOnlyDirectories= : specify file system access
- PrivateNetwork=yes: no external network access
- systemd uses namespace/seccomp, even BPF-LSM

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

Solution: nsjail

- A light-weight process isolation tool
- Making use of Linux namespaces and seccomp-bpf syscall
- Provides isolation of namespaces/filesystem/resource/
- Isolation of network service/local process
- Share the same kernel with host
- No fine-grained network policy

Solution: firejail

- It’s just like nsjail
- Restrict the running environment of untrusted application
- By using Linux namespaces, seccomp-bpf and Linux capabilities
- Can sandbox any type of process: servers, graphical applications
- Share the same kernel with host
- No fine-grained network policy

There are a lot of mechanism and solutions

- But all of them share the same kernel
- Almost(if not all) of them lack of network policy

So what sandbox do we need?

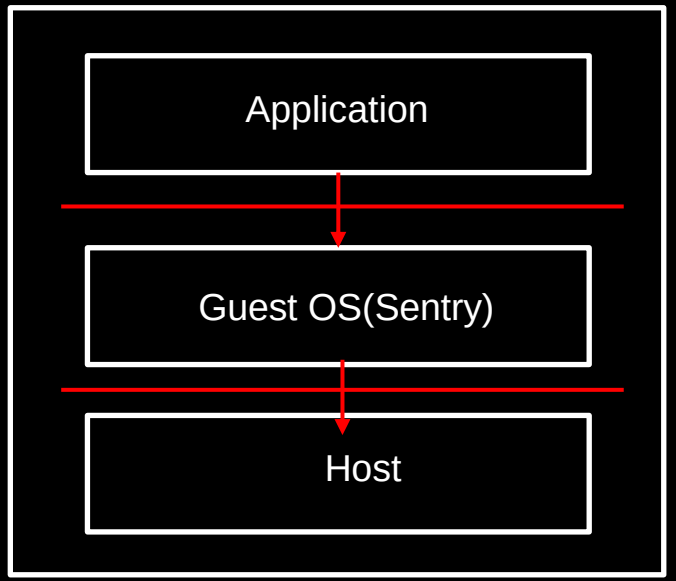
- Process restriction: defines which process can be launched
- File system access restriction: defines which file can be read/can’t be written to
- Networking access restriction: defines which ip/port/domain can be connected to
- Kernel isolation: don’t share the kernel with host

Summary: We need strong vertical and horizontal isolation

03 | gVisor overview

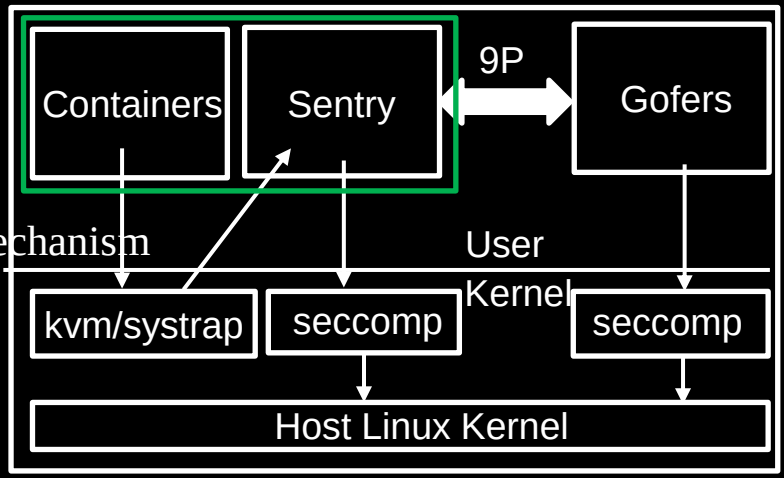
What is gVisor

- gVisor is an application kernel
- Written in Go, memory safety
- Implements a lot of Linux syscall interface, Sentry
- A lot of common Linux app can run on it, not 100%
- Implements the OCI spec



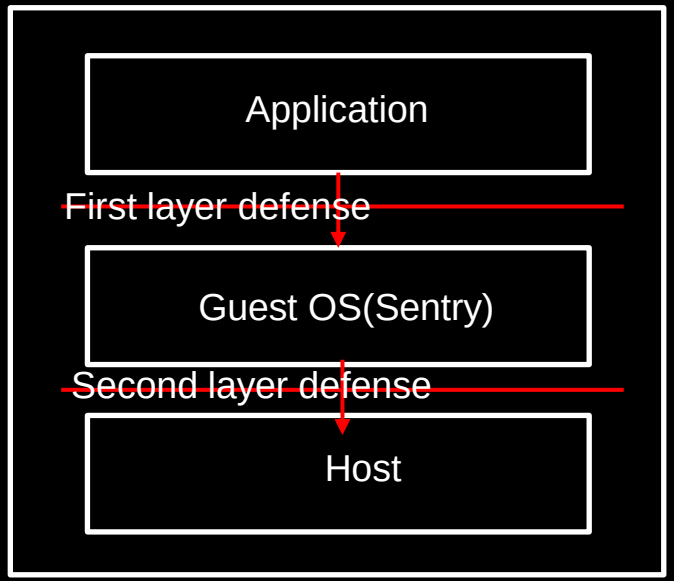
How gVisor-Defense In Depth

- Sentry: guest kernel, first layer of defense
- Use ptrace/KVM/systrap to intercept syscall
- Gofers: file system access shared
- Sentry/Gofers: both contains several security mechanism
- seccomp/capabilities/chroot/namespace/cgroup, second layer of defense



How gVisor protect the host

- First layer
  - Sentry: handle a lot of syscall request
  - Memory safety: no buffer overflow, no UAF
- Second layer
  - Secomp
  - Namespace
  - Cgroup



Why not just run sandboxed process in gVisor

- gVisor is used in cloud native/container ecosystem
- It implements OCI spec
- The OCI spec contains several security aspects for container but not all
- The OCI spec has no network-related, it’s in CNI networkpolicy
- Summary: gVisor has the vertical isolation but no horizontal isolation

gVisor hack

- gVisor is application kernel written in Go
- It’s easy to customize to meet our needs
- Let’s deny ‘ls’ execution

```
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)
```

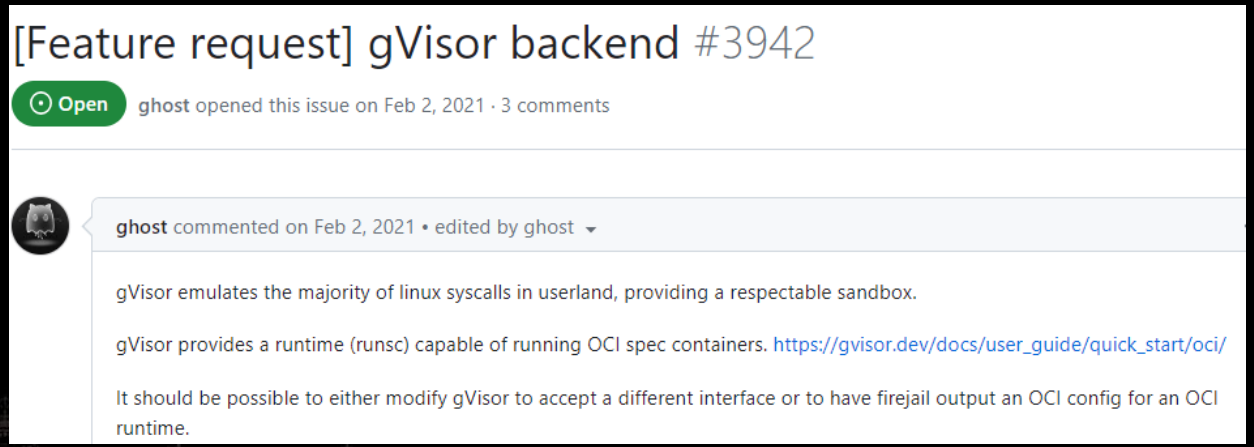
04 | Build sandbox based gVisor

Motivation

- We need a sandbox which has vertical isolation and also horizontal isolation
- Traditional solution lack of both
- gVisor implements the defense in depth and has vertical isolation
- But gVisor lack of network policy, horizontal isolation
- We need build it by ourself

But wait, can we find one

- Firejail issue
- It seems someone also want using gVisor to be an process sandbox



[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

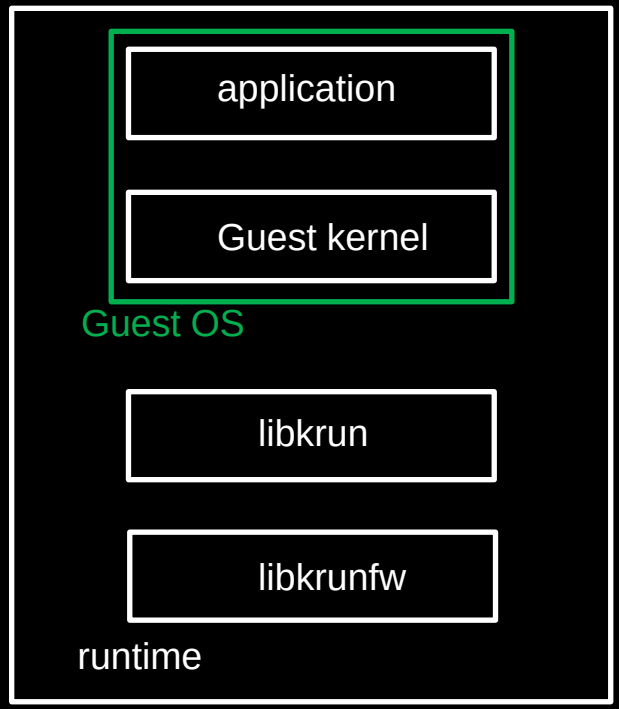
gVisor emulates the majority of linux syscalls in userland, providing a respectable sandbox.

gVisor provides a runtime (runsc) capable of running OCI spec containers. https://gvisor.dev/docs/user_guide/quick_start/oci/

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.

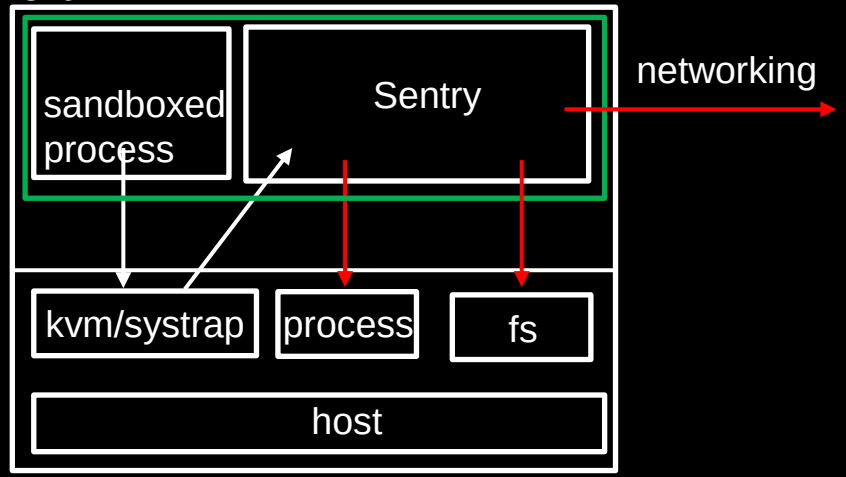
But wait, can we find one

- libkrun: a dynamic library
- That allows program to run in virtual machine
- Like gVisor, add vertical isolation
- But lack of horizontal isolation



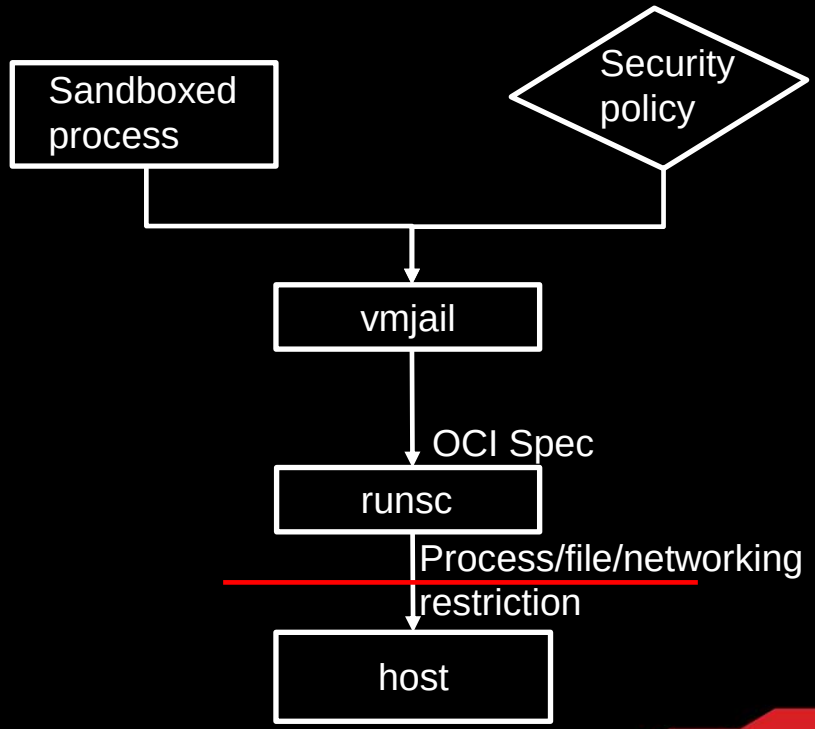
vmjail overview

- vmjail is a process-level sandbox based gVisor
- setuid binary to setup sandbox environment
- It has horizontal isolation
  - Customize the gVisor
  - define network policy
- It has vertical isolation
  - Customize the gVisor
  - define fs/process policy



vmjail architecture

- vmjail security policy
  - Process/file/networking
  - Memory/CPU
- vmjail policy->OCI spec
- runsc: start Sentry and Gofer
- Sentry: enforce security policy
  - OCI spec
  - Customization



OCI introduction

- Open Container Initiative: several spec
- Define how containers can be run
- There are several implementation of OCI
- OCI is often used as low level system in cloud native ecosystem
- OCI has several security aspects for container
- vmjail can leverage some of them

vmjail policy -> OCI spec

- vmjail policy contains all of the security policy: file, memory/CPU
- Some of them will be transferred to OCI spec
- Others are implemented in Sentry by ourself

```
"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"]
    }
}
```

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

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

File system restriction

- Define the access permission of file system
- Following policy
  - rootfs read-only: most of them can’t be write to
  - writeablePaths: The dir/file can be write to
  - maskedPaths: The dir/file that can’t be read by process

File system-OCI

- OCI has all full spec for file system access
- rootfs can be set to readonly: .root.readonly: true
- writeablePaths: set mounts
- maskedPaths: .linux.maskedPaths

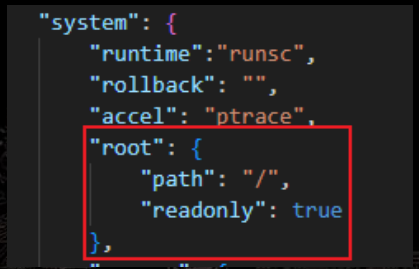
```
"root": {
        "path": "rootfs",
        "readonly": true
},
```

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

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

File system-vmjail

- vmjail can use the OCI spec directly
- vmjail create OCI spec from security policy



Network restriction

- Define the network action which can perform
- Following policy:
  - No networking at all
  - Limit outgoing IP/port
  - Limit outgoing domain name
  - Limit local listen port

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

Network-CNI

- OCI has no spec for network policy
- Container Network Interface(CNI) define the network policy
- CNI Network policy control the traffic between pods/container
- It is too heavy to use CNI

Network-vmjail

- Use gVisor host network stack (--network host)
- Modify the gVisor source code
- When run gVisor, passed it network policy
- When the application trigger network action, check whether it is allowed

Process restriction

- In most of the situation only one sandboxed program is executed
- No reverse shell, no attack tool can be run
- Executable full path as policy
- Currently it’s still in development

Process-OCI

- OCI has no spec for process restriction
- Though we can set the maskedPaths in OCI spec
- It is blacklist, we need whitelist

Process-vmjail

- vmjail policy defines the program list that can be executed
- Modify the gVisor code
- When run gVisor, passed it program policy
- When the not-in whitelist program is executed, deny it

CPU/Memory/Devices, etc

- OCI spec has spec for these resource
- vmjail can use the OCI spec directly
- CPU/Memory/Devices/Capabilities

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

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

Some issues

- Several gVisor issue
  - wget can't connect to https websites in host network mode #8156
  - statx syscall is not supported before Linux 4.11 #8229
  - gVisor upstream don’t support maskedPaths
  - gVisor cgroup delete delay
- Run as the user
  - getuid, passed to OCI spec
- gVisor require Linux 4.14
  - Allow rollback to the origin cmd in unsupport kernel

Example

- An isolation kernel
- rootfs read-only

```
'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
```

Example

- 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
drwxrwxrwt 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
```

Example-networking

- Can’t access not-in the whitelist domain

```
"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.
```

Example-networking

- Can access the whitelist domain

```
"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).
```

Example-networking

- No network at all

```
},
"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#
```

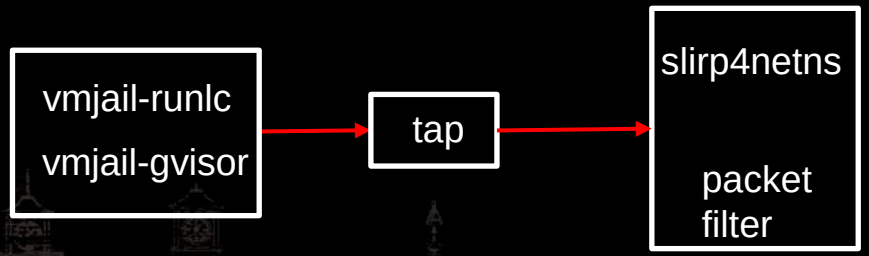
05 | The futurre

More runtime

- Currently the gVisor-based sandbox can be perfect from security perspective
- But the world is not all about security
- vmjail is suffered in some performance-critical scenes
- Some task care performance more than security
- Can we add more choices to vmjail?
- runlc, light container, based traditional tech

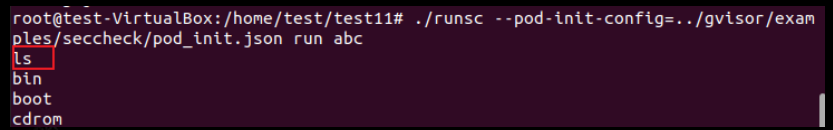
Unify network policy enforce

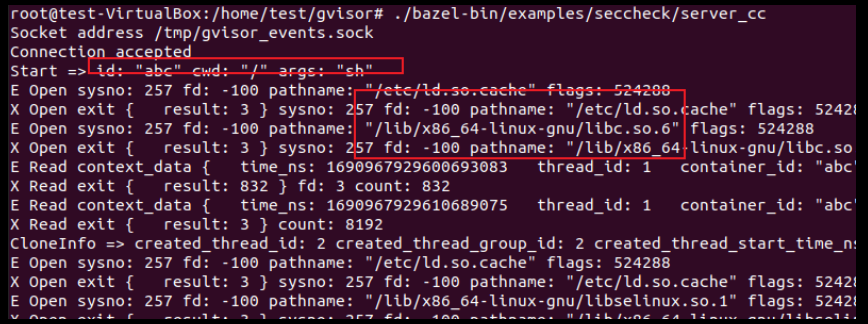
- We can add a runtime which leverages the traditional mechanism
- But we need to find a way to enforce network policy
- User space network: slirp, passt
- Packet filter in user space network stack
- Like a CNI, but more low level



gVisor for analysis sandbox

- There is another kind of sandbox which needs to monitor the behavior
- As we can see, the gVisor can inspect everything of process
- Process/Networking/File system behavior
- We can do malware analysis using gVisor





The final picture

- vmjail will have two modes
- One for enforce security policy
  - VM-based runtime: gVisor, focus security
  - namespace/cgroup-based runtime: runlc, focus performance
  - Both will have full vertical and horizontal security policy
- One for analysis program

Summary

- Currently sandbox lack some of the critical security feature
- gVisor is a full sandbox technology
- gVisor lack of several feature to be a security sandbox
- gVisor can be easily customized to meet the security needs
- We can build a powerful process-level sandbox which has strong vertical and horizontal isolation based gVisor

THANK YOU!

Li Qiang, liq3ea@gmail.com