## **Host**

**步骤**

### 获取vtzb\_proxy源码

### 获取tzdriver源码。

~~下载地址：<https://gitee.com/openeuler/itrustee_tzdriver>~~

~~代码分支：master~~

### 获取libboundscheck源码。

下载地址：<https://gitee.com/openeuler/libboundscheck>

代码分支：master

### 将下载、解压缩后libboundscheck源码放置在“vtzb\_proxy/”目录下，并修改文件夹名称为libboundscheck

### 编译vtzb\_proxy

（可选）按实际串口数量修改serial\_port.h，一个VM对应一个虚拟串口

#define SERIAL\_PORT\_NUM 15

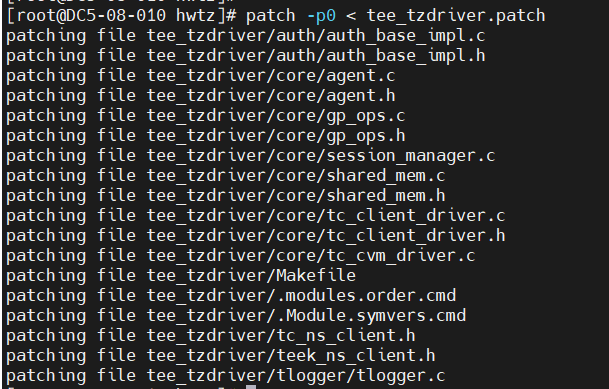
cd vtzb\_proxy && make  
sudo cp ./vtz\_proxy /usr/bin/vtz\_proxy

### Patch Application

进入目标目录 tzdriver的父级目录，确保补丁文件和目标目录在同一级目录下

patch -p0 < itrustee\_tzdriver.patch

打印信息：



### 编译tzdriver

cd tzdriver && make

### 加载tzdriver.ko

sudo insmod tzdriver.ko

## **QEMU**

### 获取qemuv5.0.0源码

git clone -b v5.0.0 https://git.qemu.org/git/qemu.git

### Patch Application

进入目标目录 qemu 的父级目录，确保补丁文件和目标目录在同一级目录下

patch -p0 <qemu.patch

打印信息：

[vmuser@test]$ patch -p0 <qemu.patch  
patching file qemu/hw/char/tc\_ns\_client.h  
patching file qemu/hw/char/tee\_client\_constants.h  
patching file qemu/hw/char/tee\_client\_list.h  
patching file qemu/hw/char/tee\_client\_type.h  
patching file qemu/hw/char/virtio-console.c  
patching file qemu/include/monitor/monitor.h  
patching file qemu/monitor/misc.c

### 编译qemu

cd qemu  
mkdir build  
cd build  
../configure --target-list=aarch64-softmmu  
make -j  
​

### 若已经安装虚拟机可直接修改虚拟机vm.xml配置文件（请备份vm.xml文件）

<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>  
   .......  
   <devices>  
 .......  
   </devices>  
 <!-- 添加一个虚拟串口，映射到host路径/tmp/vm\_vtzb\_sock0 --!>   
   <qemu:commandline>  
     <qemu:arg value='-chardev'/>  
     <qemu:arg value='socket,path=/tmp/vm\_vtzb\_sock0,server=on,wait=off,id=vm03\_vtzb\_sock'/>      
     <qemu:arg value='-device'/>  
     <qemu:arg value='virtio-serial'/>      
     <qemu:arg value='-device'/>  
     <qemu:arg value='virtserialport,chardev=vm03\_vtzb\_sock,name=vtzf\_serialport0'/>     
     <qemu:arg value='-monitor'/>  
     <qemu:arg value='telnet:127.0.0.1:5556,server,nowait'/>     
   </qemu:commandline>  
   <seclabel type='dynamic' model='dac' relabel='yes'/>  
</domain>

path按照/tmp/vm\_vtzb\_sock0，/tmp/vm\_vtzb\_sock1，/tmp/vm\_vtzb\_sock2且接着上一个vm.xml的path有序递增；修改id与chardev 一致且唯一；name 恒为vtzf\_serialport0；

#### 从零构建虚拟机（忽略）

1. 使用qemu-img工具的create命令，创建镜像文件

qemu-img create -f qcow2 openEuler-image.qcow2 50G

2）配置vm.xml文件

<domain type='kvm'>  
   <!-- 虚拟机名称 -->  
   <name>openEulerVM</name>  
   <!-- 虚拟机内存 -->  
   <memory unit='GiB'>32</memory>  
   <!-- 虚拟处理器的个数 -->  
   <vcpu>16</vcpu>  
   <os>  
       <type arch='aarch64' machine='virt'>hvm</type>  
       <!-- 这里是arm架构的 -->  
       <loader readonly='yes' type='pflash'>/usr/share/edk2/aarch64/QEMU\_EFI-pflash.raw</loader>  
       <!-- 指定nvram文件路径，用于存储UEFI启动配置 保证文件名不冲突即可 -->  
       <nvram>/var/lib/libvirt/qemu/nvram/openEulerVM.fd</nvram>  
   </os>  
   <features>  
       <acpi/>  
       <gic version='3'/>  
   </features>  
   <cpu mode='host-passthrough'>  
   </cpu>  
   <iothreads>1</iothreads>  
   <clock offset='utc'/>  
   <on\_poweroff>destroy</on\_poweroff>  
   <on\_reboot>restart</on\_reboot>  
   <on\_crash>restart</on\_crash>  
   <devices>  
       <emulator>/usr/libexec/qemu-kvm</emulator>  
       <disk type='file' device='disk'>  
           <driver name='qemu' type='qcow2' iothread="1"/>  
           <!-- 镜像路径 -->  
           <source file='/home/images/openEuler-image.qcow2'/>  
           <target dev='vda' bus='virtio'/>  
           <boot order='1'/>  
       </disk>  
       <disk type='file' device='cdrom'>  
           <driver name='qemu' type='raw'/>  
           <!-- 安装系统所需光驱 -->  
           <source file='/home/iso/openEuler-22.03-LTS-SP1-aarch64-dvd.iso'/>  
           <readonly/>  
           <target dev='sdb' bus='scsi'/>  
           <boot order='2'/>  
       </disk>  
       <interface type='bridge'>  
           <!-- 网桥 -->  
           <source bridge='br0'/>  
           <model type='virtio'/>  
       </interface>  
       <console type='pty'/>  
       <video>  
               <model type='virtio'/>  
       </video>  
       <controller type='scsi' index='0' model='virtio-scsi'/>  
       <controller type='usb' model='ehci'/>  
       <input type='tablet' bus='usb'/>  
       <input type='keyboard' bus='usb'/>  
       <!-- port 为vnc连接端口，passwd 为连接密码 -->  
       <graphics type='vnc' port='5901' listen='0.0.0.0' passwd='passwd1234!'/>  
   </devices>  
 <qemu:commandline>  
 <!-- 添加一个虚拟串口，映射到host路径/tmp/vm\_vtzb\_sock0 -->  
     <qemu:arg value='-chardev'/>  
     <qemu:arg value='socket,path=/tmp/vm\_vtzb\_sock0,server=on,wait=off,id=vm03\_vtzb\_sock'/>      
     <qemu:arg value='-device'/>  
     <qemu:arg value='virtio-serial'/>      
     <qemu:arg value='-device'/>  
     <qemu:arg value='virtserialport,chardev=vm03\_vtzb\_sock,name=vtzf\_serialport0'/>     
     <qemu:arg value='-monitor'/>  
     <qemu:arg value='telnet:127.0.0.1:5556,server,nowait'/>     
   </qemu:commandline>  
   <seclabel type='dynamic' model='dac' relabel='yes'/>  
</domain>

使用远程证明请在配置文件中添加字段

<vsock model='virtio'>

<cid auto='no' address='9'/>

</vsock>

address唯一且不小于3.

### 定义并启动虚拟机

virsh define vm.xml  
virsh start openEulerVM  
/\* 若新安装，用VNC登录安装 \*/

## **VM**

CATA依赖

pip3 install pycryptodome

### 获取itrustee\_client源码。

~~下载地址：<https://gitee.com/openeuler/itrustee_client>~~

~~代码分支：master~~

### 获取vtzdriver与virtio(5.10内核)源码。

### 获取libboundscheck源码。

下载地址：<https://gitee.com/openeuler/libboundscheck>

代码分支：master

### 将下载、解压缩后libboundscheck源码放置在“vtzdriver”目录下，并修改文件夹名称为libboundscheck

### 编译itrustee\_client。

~~cd itrustee\_client && make~~

### 部署itrustee\_client。

~~cp -rf dist/\\*.so /usr/lib64 && ldconfig  
cp -rf dist/teecd /usr/bin  
cp -rf dist/tlogcat /usr/bin  
\# 可选  
cp -rf dist/tee\_teleport /usr/bin  
cp -rf dist/agentd /usr/bin~~

### 编译virtio\_console.ko并加载（仅5.10内核，openEuler22.03）

先卸载rmmod virtio\_console，必须在加载vtzdriver前卸载！

rmmod virtio\_console

cd ./virtio/char

make

insmod ./virtio\_console.ko

### 编译vtzdriver并加载vtzfdriver.ko

若kernel路径不正确，请自行替换Makefile

KERN\_VER = $(shell uname -r)

KERN\_DIR = /lib/modules/$(KERN\_VER)/build

cd vtzdriver && make

sudo insmod vtzfdriver.ko

**vtzfdriver加载后不可卸载。**

## **RUN**

Host重启后重新加载驱动，host需运行teecd加载kunpeng\_sec\_drv.sec，请确保kunpeng\_sec\_drv.sec已放入指定路径。

host重启后需要卸载二级证书才能升级为CCOS，升级后请重新加载二级证书。所有第三方CATA运行均需要二级证书签名后编译TA。

请在升级CCOS并导入二级证书后在host侧运行teecd，再运行rsa\_demo验证host环境是否正常。

host:

sudo /usr/bin/teecd

//卸载二级证书

sudo /usr/bin/certmanager destroy

//升级机密OS

..............................

//完成升级

//重启teecd加载二级证书

sudo /usr/bin/teecd

sudo /usr/bin/certmanager import second.der

//启动vtz\_proxy代理

sudo /usr/bin/vtz\_proxy

VM：按照host的相同路径放置kunpeng\_sec\_drv.sec

sudo /usr/bin/teecd

sudo /vendor/bin/testcase1 [1-9]

## 远程证明

### 环境搭建

1. 参照《Kunpeng BoostKit 23.0.RC2 机密计算TrustZone套件 特性指南》在host编译部署 QCA lib 和 QTA、在VM编译部署 qca-report 和 QTA-report；注意HOST与VM不同的编译选项。建议在host上预先测试远程证明，熟悉操作并确保环境正常。
2. 确保VM配置文件已添加字段

<vsock model='virtio'>

<cid auto='no' address='9'/>

</vsock>

1. 获取qca\_host\_server\_demo、qca\_client\_demo、qca\_guest\_demo源码。

qca\_host\_server\_demo、qca\_client\_demo置于Host侧，

qca\_guest\_demo置于VM侧；

1. 获取/itrustee\_sdk-1.0

git clone -b itrustee\_sdk-1.0 <https://gitee.com/openeuler/itrustee_sdk.git>

1. 获取cJSON

git clone https://github.com/DaveGamble/cJSON.git

1. 将cJSON分别拷贝至qca\_host\_server\_demo、qca\_client\_demo、qca\_guest\_demo目录下

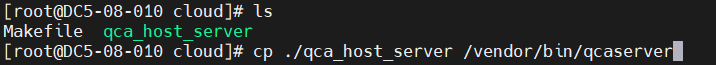
如图所示

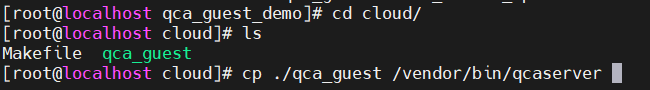






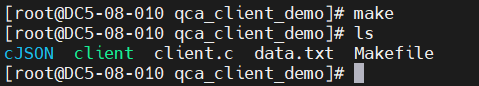
1. 将qca\_host\_server\_demo、qca\_guest\_demo拷贝至itrustee\_sdk/test/CA/
2. Host与VM分别进入qca\_host\_server\_demo、qca\_guest\_demo 下cloud目录，执行make，将二进制文件拷贝至 /vendor/bin/qcaserver





1. 编译部署qca\_client\_demo

进入qca\_client\_demo目录，执行make。无需拷贝至特定目录。

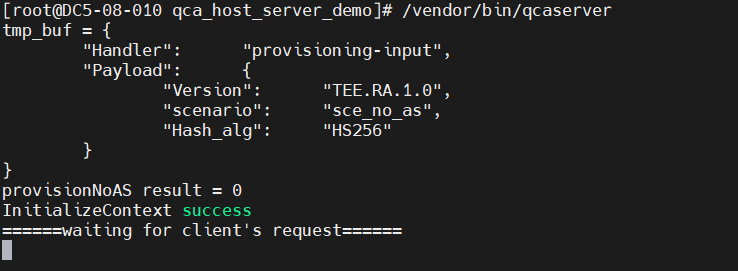


1. Host执行 modprobe vhost\_vsock

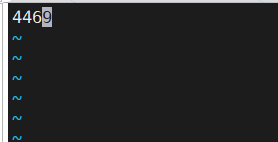
**执行远程证明**

1. Host运行/usr/bin/teecd 与/usr/bin/vtz\_proxy
2. VM运行/usr/bin/teecd
3. Host运行 /vendor/bin/qcaserver

若正常，将显示

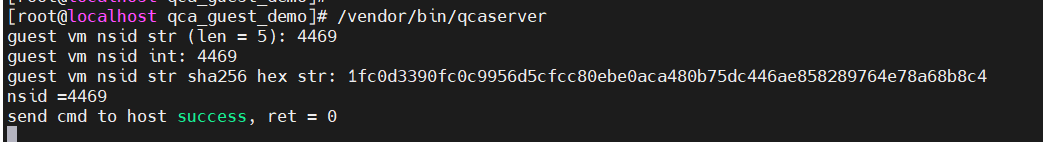


1. VM查看vim /tmp/qca\_nsid ，确认qemu的pid

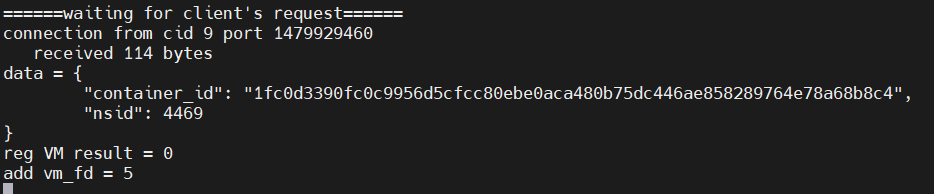


运行 /vendor/bin/qcaserver

若正常VM打印

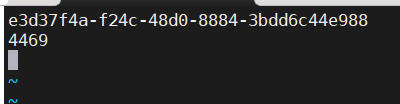


Host打印



1. Host修改qca\_client\_demo目录下data.txt，第一行为待验证TA的uuid，第二行为指定VM的pid





1. VM运行待验证的demo，确保证证明期间demo不退出
2. qca\_client\_demo目录下执行./client 127.0.0.1

若正常，将打印report信息。

