COL732: Project Report

INTRODUCTION

This is a collaborative project where we aim to create a cloud service for teaching and evaluations using vmm-reference. Using this service, instructors can upload assignments and a grading rubric by means of automated tests, the students can implement their solutions and submit them, and the TAs can evaluate the submissions.

As a smaller part of this project, we had to implement ssh functionality so that students can access their VM. Our goal is to provide students with various libraries so as to ease their coding experience. We will also focus on how to run commands from Hypervisor to VM.

PROJECT IMPLEMENTATION DETAILS

IP Assignment

How will a VM get its IP address?

```
# Assigns IP address to the VM
ip a add {{ip}}/24 dev eth0
ip link set eth0 up
```

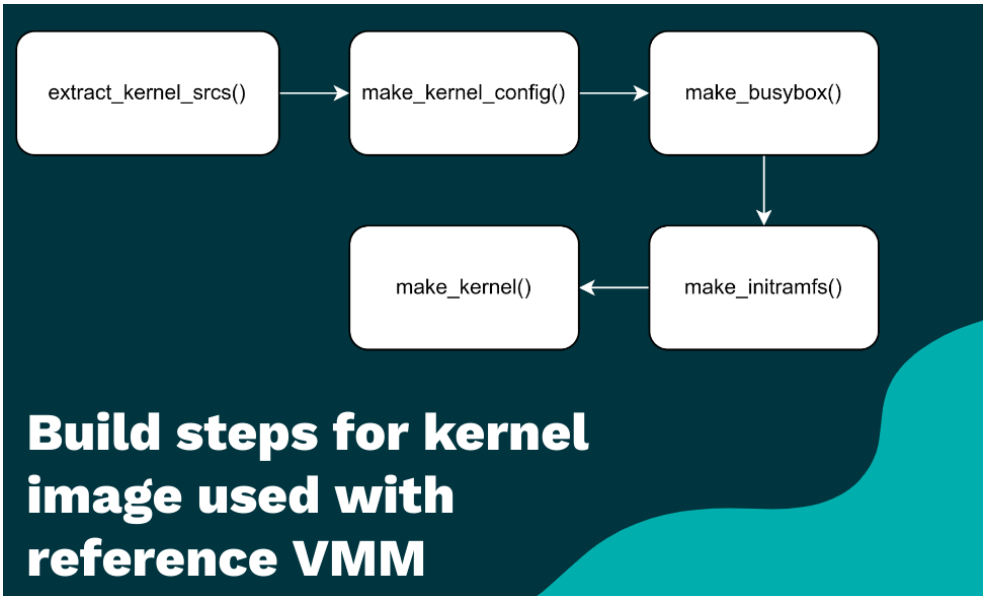
- `VM ID` to `IP address` bijection function
- IP address of Hypervisor (with ID=x) = 192.168.{200+x}.1
- IP address of VM = 192.168.{200+x}.2
- Tap device = vmtap{100+x}
- Commands executed in VM from hypervisor as we need IP address as a parameter
- One hypervisor per VM and separate tap device for each hypervisor, and thus each VM.

Base Image with Additional Libraries

Building a custom kernel busybox image

- Step 1 : Acquire linux kernel sources and relevant config

    - Download kernel sources and copy base kernel config
- Step 2: Make the initramfs in the kernel directory
    - `initramfs` is a root filesystem that is embedded into the kernel and loaded at an early stage of the boot process.
    - a) Download, extract and build statically linked busybox
    - b) Copy busybox and statically compiled binaries of all required functionalities (Dropbear, Python 3) into initramfs directory
- Step 3: Put them together and build the kernel image



Dropbear for SSH

Why Dropbear?

- Provides both SSH server and client utilities.
- No external dependencies, so typically useful for embedded Linux systems.
- Since busybox VM image run with a smaller environment, thus Dropbear is the right choice for SSH utility on busybox.

Setup for Dropbear

- Download dropbear from here.
- Configure dropbear and enable static compilation using command `./configure --enable-static`
- Adding dropbear to base VM image
    - Install dropbear while building busybox base image.

    - `make PROGRAMS="dropbear dbclient dropbearkey dropbearconvert scp" install DESTDIR={path to initramfs}`
- Place dropbear executable files in /bin directory of the base VM image.
    - In init code of image add following commands
        - `cp /dropbear/usr/local/sbin/* /bin/`
        - `cp /dropbear/usr/local/bin/* /bin/`

Start Dropbear SSH server

- Add following commands in the init code of VM image

```
# Generate Keys

dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key > dump.txt
dropbearkey -t dss -f /etc/dropbear/dropbear_dss_host_key > dump.txt
dropbearkey -t ecdsa -f /etc/dropbear/dropbear_ecdsa_host_key > dump.txt
dropbearkey -t ed25519 -f /etc/dropbear/dropbear_ed25519_host_key > dump.txt

# Create configuration file for SSH server

echo "NO_START=0" > /etc/default/dropbear
echo "# the TCP port that Dropbear listens on" >> /etc/default/dropbear
echo "DROPBEAR_PORT=22" >> /etc/default/dropbear
echo "# any additional arguments for Dropbear" >> /etc/default/dropbear
echo "DROPBEAR_EXTRA_ARGS="-s"" >> /etc/default/dropbear
echo "# specify an optional banner file containing a message to be" >> /etc/default/dropbear
echo "# sent to clients before they connect, such as "/etc/issue.net"" >> /etc/default/dropbear
echo "DROPBEAR_BANNER="SSH functionality working "" >> /etc/default/dropbear
echo "# RSA hostkey file (default: /etc/dropbear/dropbear_rsa_host_key)" >> /etc/default/dropbear
echo "DROPBEAR_RSAKEY="/etc/dropbear/dropbear_rsa_host_key"" >> /etc/default/dropbear
echo "# DSS hostkey file (default: /etc/dropbear/dropbear_dss_host_key)" >> /etc/default/dropbear
echo "DROPBEAR_DSSKEY="/etc/dropbear/dropbear_dss_host_key"" >> /etc/default/dropbear

# Create device folder

mkdir /dev/pts
mount -t devpts /dev/pts /dev/pts
mdev -s

# Create passwd file in \etc folder to create users
touch /etc/passwd
```

- After IP has been assigned to dropbear, start server using
    - `dropbear -E > log.txt`
- Add user and set password using
    - `adduser -D {username}`
    - `echo '{username}:{password}' | chpasswd`
- SSH server is ON!

Run Commands from Hypervisor to VM

Run commands from Hypervisor to VM

- VM traps into hypervisor during IoIn, IoOut, MemRead, and MemWrite
- At this point, we can inject desired commands into VM
- At boot time, VM outputs '/ # ' and then waits for the user to enter command
- So, we used a state machine to identify the instant when VM printed '/ # '.
- After this instant, when VM traps for IoIn, we inserted the required command

Pattern to Input a Command

- We entered arbitrary characters and used print statements in VcpuExit::IoIn to identify how it is being read
- It reads 196, 97 then the ascii value of the character and then 96, 96
- Using the above pattern, we inserted the characters of the command and when VM sees '\n' it executes it
- Then it again outputs '/ # ' and waits for input
- So, we can run arbitrary number of commands

Run arbitrary Script at Launch

- Pass script as command line argument while launching VM
- Instead of running hard-coded commands, read the script and run commands one by one
- Modified vmm_reference source code
- Extra argument starter_file while launching VM
- Needed the filename in KvmVcpu
    - Starting with argument parser, decided the flow of starter_file
    - Changed the structures according to the flow

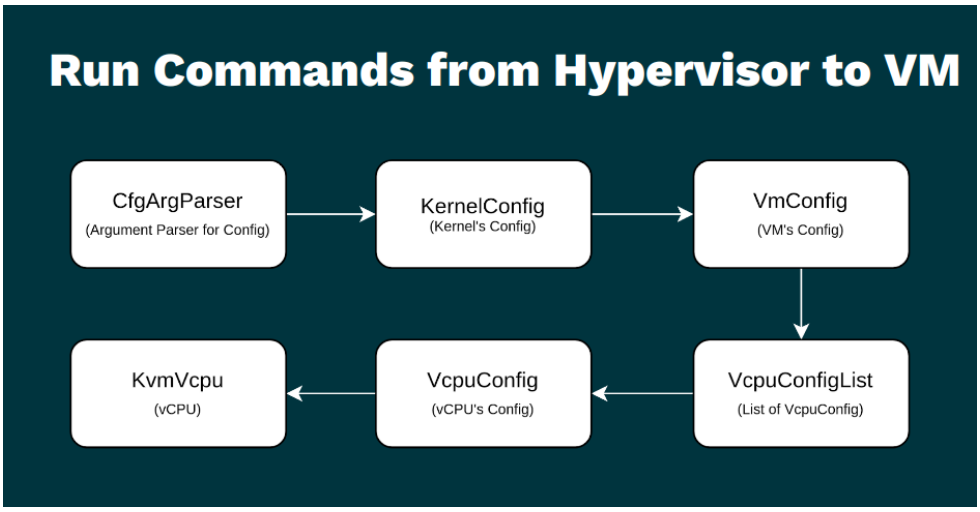
Passing script as CL argument

- So, we add starter_file to kernel's config and argument
    - `./vmm_reference --kernel path=base_image,starter_file=filepath`
- Starter file can have arbitrary commands to run separated by newline
- From KernelConfig, it reaches VcpuConfig where KvmVcpu can use it
- Now we just need to run the script from hypervisor

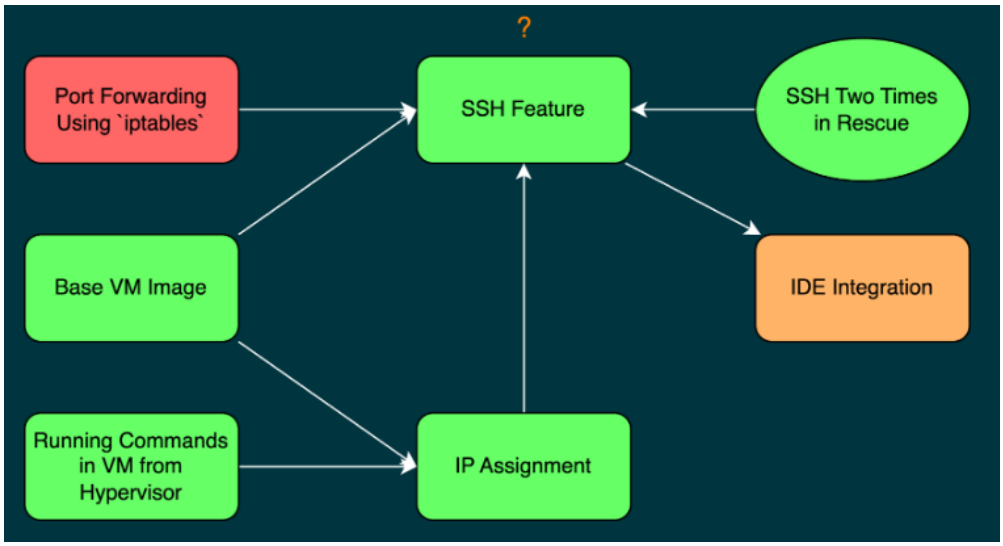
From Script to VM

- Used the idea discussed to run commands for hypervisor to VM
- Modified it to make it generic for any list of commands

- Read the file and passed the commands
- AND WE ARE DONE!



DESIGN DOCUMENT



CODE LINK

The src folder of vmm-reference was changed for ‘Run Commands from Hypervisor to VM’ (as mentioned above).

Here is the new src folder. The files changed are below-

- src/main.rs
- src/api/src/lib.rs

- src/vmm/src/lib.rs
- src/vmm/src/config/{builder.rs, mod.rs}
- src/vm-vcpu/src/vm.rs
- src/vm-vcpu/src/vcpu/mod.rs

make_busybox.sh file was updated for dropbear configuration

CHALLENGES

- Not able to configure iptables to implement port forwarding

```
iptables -t nat -A PREROUTING -p tcp --dport 8080 -j DNAT --to-destination 192.168.241.1:4096

iptables -t nat -A POSTROUTING -p tcp --dport 4096 ! -s 127.0.0.1 -j MASQUERADE

iptables -A FORWARD -p tcp --dport 4096 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
```

- Scanning the code of vmm-reference for ‘Run Commands from Hypervisor to VM’
- SSH installation inside the VM
- Dropbear configuration

FUTURE SCOPE

- Figure out `iptables`
- Improve usage of tap devices, IP assignment will also improve as a consequence
- Finish IDE integration
- Make support for running other than improvised busybox kernel images
- Install basic libs like curl and apt for installation of libs from inside the VM
- TAs may want to run some command in all the launched VMs
    - Currently commands can be run only at launch
    - Can generalize

EXPECTED & ACHIEVED OKRs

Expected

- Building the base VM image and providing sufficient pre-installed libs
- Enable students to do SSH into their VMs
- VS Code IDE support using SSH
- Flexible and Scalable autograding mechanism

Achieved

- Building the base VM image and providing sufficient pre-installed libs
- Running commands from Hypervisor in VM at VM start
- IP Assignment
- Enable students to SSH (and SCP) into their VMs (?)
- VS Code IDE support using SSH
- ~~Flexible and Scalable autograding mechanism~~ (Transferred to Anti-Cheating team)

MILESTONES

- Students can SSH and code inside the VM in python
- Can install more libraries if needed in base image
- Starter script is very powerful
- Upgrade running commands in VM from hypervisor feature to run the commands at any time in the running VM instead of only at the start
- Autograder can scp submission files from the VM (SCP accidentally works only from the host)

CONTRIBUTION OF EACH MEMBER (100 Tokens)

- Abhay Pratap Singh Rathore, 2019CS50414
    - 10: Mostly the managing part + IP Assignment + Failed attempt of IPTables
- Aditya VNS 2019CS50471
    - 21: Figured out how to run commands from hypervisor to VM
- Akshat Gupta, 2019CS50418
    - 15: Coded the functionality of running starter script in VM
- Arpit Chauhan, 2019CS10332
    - 15: Installed libraries inside base VM image
- Himanshi Ghai, 2019CS50433
    - 18: Figured out how to compile bzimage with additional libraries. Setup of dropbear for ssh client and server.
- Prakhar Aggarwal, 2019CS50441
    - 21: Figured out how to compile bzimage with additional libraries. Setup of dropbear for ssh client and server.