RESEARCH ARTICLE OPEN ACCESS

Domain Analysis and Feature Identification of Virtualization in Debian Linux Using Type II Hypervisor – Virtual Box

Dr. Vinit A. Sinha [1], Dr. Vilas M. Thakare [2]
[1] Assistant Professor, PGDCA, PRMITR Badnera
[2] Ex- Head, Professor, SGBAU, Amravati

ABSTRACT
Research outlines network security regarding linux operating system, specially ubuntu 22.04 and kali linux 2021. Another aspect of security implementation of virtualization techniques is to maintain security level of operating system in guest system as compared to host system. As discussed by Junior et al (2018) about self-adaptive firewall, which include high level policies, vulnerability scanning and detection and firewall rule application engine. This work describe KVM networking with lib-virt, vulnerability scanning and intrusion detection techniques with implementation of snort and scanning with Nikto. Furthermore in research work, feature extraction was explored and it involves deployment and implementation of python script which are executed in virtual environment on guest system. This research work includes installation and configuration of various tools like lib-virt, KVM instances, virt-manager and other security aspects.
Keyword: -Kali Linux, Ubuntu, KVM, virtual box, firewall.

I. INTRODUCTION

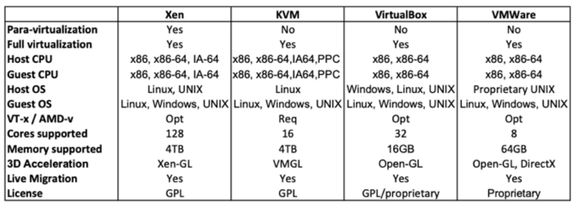
While implementation of virtualization security techniques in linux network environment, hardware and software platform have a prominent place. During execution of security techniques lots of factor as hardware / software requirement, KVM implementation, network and vulnerability scanning are evaluated at extinct level. Here once again firewall technology as security components adds its value in network security management. While handling firewall its configuration is setting up in both GUI and CUI manner. After setting firewall, KVM had another aspect for setting virtualization in network environment.

II. RELATED WORK

This section discusses about the literature survey of different researchers in the field of virtualization and network security. The researcher Ganji et al. [1] examined suitable infrastructure for the linux OS. They also mentioned linux system security requirements in server network security. Patil et al. [2] proposed hypervisor level distributed network security (HLDNS) framework which is deploy to monitor VM related network traffic for intrusion detection. Li et al. [3] proposed framework, which divides network security into five stages which are as factor acquisition, model representation, measurement establishment, solution analysis and situation prediction. Bock et al. [4] explain real time hypervisor based on Xvisor for delivering secure and separated environment for virtualized system. Compastie et al. [5] describe comparison between different virtualization models to build architecture of cloud infrastructure for analyzing network security issues. Yan et al. [6] verified effectiveness of system using client-side distributed energy storage demo project for improving system load characteristics. Potdar et al. [7] describes server virtualization, which provides a platform to run different OS services. They describe performance evaluation of docker container and virtual machines using effective tools. Bahn et al. [8] explains separations of VM for managing unnecessary network traffic.

III. METHODOLOGY

Many IT organization hold virtualization technology enthusiastically for improving efficiency of their resource utilization. Virtualization allows all system administrators to run many guest system (VM) on single host system (Physical). This techniques allows guest system to share resources from host system in terms of memory, storage, CPU core and other resources. Virtualization allows administrators to reallocate resources rapidly on demand basis or as per need in shift. Following **fig. 1.1** shows comparison between Xen, Virtual Box, VMware and KVM.



	Xen	KVM	VirtualBox	VMWare
Para-virtualization	Yes	No	No	No
Full virtualization	Yes	Yes	Yes	Yes
Host CPU	x86, x86-64, IA-64	x86, x86-64,IA64,PPC	x86, x86-64	x86, x86-64
Guest CPU	x86, x86-64, IA-64	x86, x86-64,IA64,PPC	x86, x86-64	x86, x86-64
Host OS	Linux, UNIX	Linux	Windows, Linux, UNIX	Proprietary UNIX
Guest OS	Linux, Windows, UNIX	Linux, Windows, UNIX	Linux, Windows, UNIX	Linux, Windows, UNIX
VT-x / AMD-v	Opt	Req	Opt	Opt
Cores supported	128	16	32	8
Memory supported	4TB	4TB	16GB	64GB
3D Acceleration	Xen-GL	VMGL	Open-GL	Open-GL, DirectX
Live Migration	Yes	Yes	Yes	Yes
License	GPL	GPL	GPL/proprietary	Proprietary

Fig. 1.1 . Comparison between Virtualization methods

Above figure shows various virtualization methods, all of them supports full virtualization. Today for x86 platform on virtualization, it is common practice to support this type of hardware and software support. Originally Xen is launched as paravirtualization software or VMM. But in advanced to this full virtualization doesn't need to manipulate guest kernel at all. In this full virtualization guest and host CPU are listed properly and supported by x86 / x64 or AMD 64 guest cores.

For full virtualization Xen and KVM support Itanium 64 processor architecture. Due to QEMU dependencies and instruction point of view KVM is only virtualization method supported by Intel VTX or AMD – V. As advanced techniques and both hardware and software support, virtual box and VMware are introduced to managed all kind of instruction set.

Another aspect of implementation of these all virtualization method is license support for application stable support under high performance computing (HPC). Here KVM, Virtual Box and XEN are provide there service under GNU public license (GPL) version 2. It means they open to utilize and modified by anyone within open source community at another site virtual box was launched in GPL license category , recently it acquired more features on platform of virtualization and supported by oracle. On another side VMware had different class and limited licensing skim, which restricted user from widely spread up of performance footprints with taking prior approval. So it is better to go with open source , in this research virtual box 6.1 is effectively used for making virtualization of LINUX and MAC OS as well as windows as a host system. Virtualization domain consists of modelling structure in virtual environment with network support. While using virtualization in network system it add security perspective to host system for managing guest machine and their hosting sites. Different types of virtual system in domain of virtualization contains VMWARE, XEN and generic virtual images.

IV. FEATURE IDENTIFICATION

1. Utilization of hardware –

By minimising requirement of physical devices (hardware) or system , virtualization cut off list used for setup of network server. This include maintenance cost , power consumption , minimum number of hardware devices as well as network devices. It helps to allocate CPU, memory space and RAM, customization in just a second.

2. Recovery in disaster situation –

In critical situation like hackers attack or any data leak and exploitation situation network serves are recovered easily if they base on virtualization. By using snapshot feature users can get back to work where they left in crucial situation in less time. It also help to backup of server data in case of maintenance situation.

3. Increase availability of serving more –

Virtualization in network server hosting, help to increase uptime of sever for providing better services in network. As it provides unique features rather than physical server, it increases productivity of guest system in web application scenario. It helps to extent following mentioned capabilities as ,

- Distributed resource scheduling
- Live Migration
- Fault Tolerance
- Storage Memory migration
- High Availability of Resources

4. Fast Set up Of data Centre –

Virtualization technology helps to move from physical server to virtual network which helps to reduce hardware consumption. These result out to save energy, clean up air and carbon footprint. This action minimize pollution in environment. Virtualization Technology (VT) helps to deploy server fast by cloning images , VMs and network prototype. This activity does not need any purchase order or physical paper copy. Virtualization technology helps to configure server in terms of

OS, CPU core and needy application. Finally as it minimises H/W and supporting devices it creates useful space in server room as well as in data centre.

5. Shifting Local Lab on cloud in single day –

Hypervisors in virtual System allow to manage N/W tools to export all network service to cloud system in local networking LAB. For this goal Virtualization Infrastructure is needed.

6. Distributed network Acquisition –

VT complete there on the basis of distributed network, so all different applications whose recommended hardware requirements changes on demand are managed by virtual instances. These instances are vary as per application requirement and can be minimized to handle single priority task execution by post server.

V. HARDWARE REQUIREMENT FOR VIRTUALIZATION

For setting up. Server system in virtual environment following configuration of system is required

1. CPU – Minimum Core i3 - 2nd generation
   Recommended Core i5 – 5th generation
   (64- bit – 86 intel) / AMD Athlon 64Fx 2.4Ghz – 3.1 Ghz Processor
2. RAM – 4GB Minimum / 8 GB or 16 GB recommended
3. HDD – 30 GB for virtual hypervisor , 15 GB for each Guest OS.
4. Graphics Card – 2 GB NVIDIA for running graphics application.
5. Swap memory – 4 GB Swap for improving Primary memory performance.
6. PORT – USB port 2.0 – 3.1 for boot from USB and Other devices.
7. CD/DVD drive – for installing extended packages and repository.

For setting network for server up in virtual environment, following configuration on network is to be fulfilled .

Network setup consist of following VM and Components –

1. Kali Linux 2021.4
2. Ubuntu *Jammy Jellyfish* - 22.04
3. MR-Robot
4. Metasploitable 2
5. Virtual Box DHCP server
6. Vbox network Isolation
7. Microsoft Windows 10

After successful setup of network a desktop is appeared as shown in **fig. 1.2**

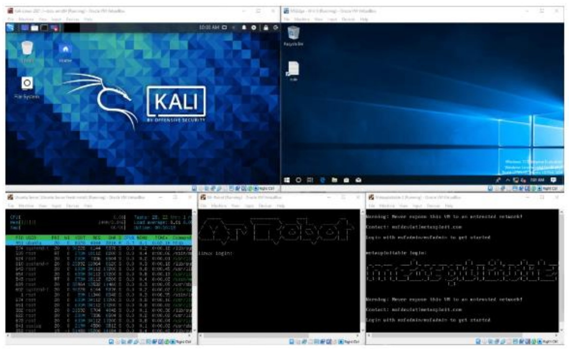


Fig. 1.2. Virtual Desktop of different OS

For a network a setup , High configured laptop or desktop (i3 / i5 processor, minimum 16 GB RAM , 100 GB HDD / SSD) Fibre optics Internet connection) is required.

VI. SOFTWARE REQUIREMENT FOR VIRTUALIZATION

6.1. Host system –

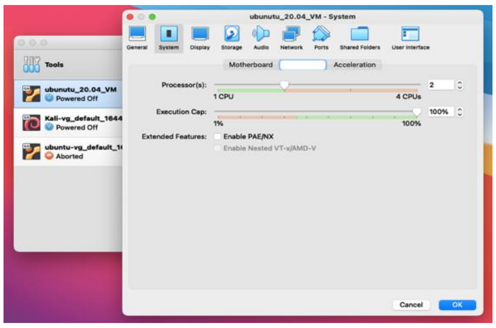
For providing platform to guest system, host system must be properly configured and should hold recommended OS software. Host OS must be based on x86 - x64 bit processor.

Research works on Linux server. It may be any GNU/ Linux OS. I can be from any class from CentOS, Debian, Red HAT or cloud Linux. From windows class it can be Windows Server 2016 to Windows Server 2022. With this flavours of operating system host system should contain apache as web server and Linux GNU license. With this requirement for Host system somethings should be avoided by host machine i.e. resource consuming applications, which may let down performance of guest systems (VMs).

Host system should be supported by DNS server, mail server for smooth working of network server system.

6.2. Virtual Box (Guest OS platform) –

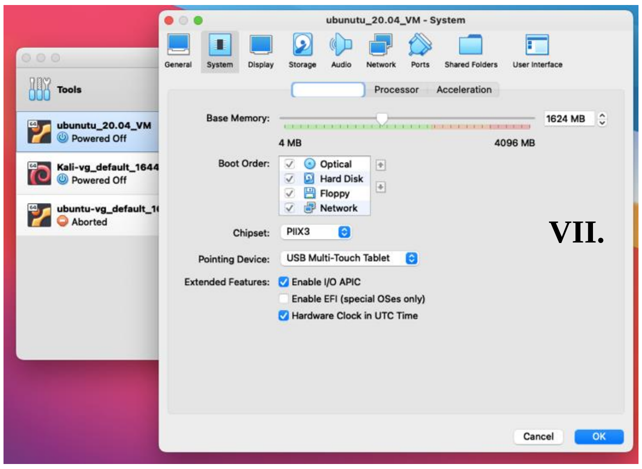
Setting up virtual box for virtual environment some elements are need to be setup. Following figures shows their configuration one by one.


ubunutu_20.04_VM - System

Processor(s): 2 (1 CPU – 4 CPUs)
Execution Cap: 100% (1% – 100%)
Extended Features: ☐ Enable PAE/NX
☐ Enable Nested VT-x/AMD-V

General System Display Storage Audio Network Ports Shared Folders User Interface
Motherboard Acceleration
Cancel OK

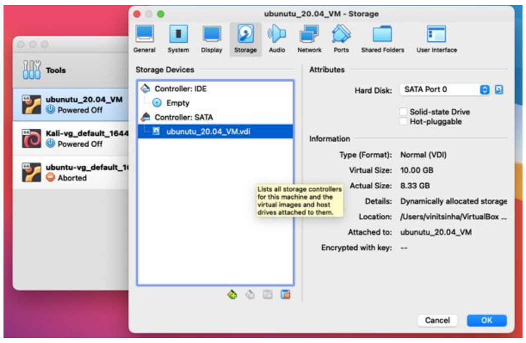
Fig. 1.3 Processor setup


ubunutu_20.04_VM - System

Base Memory: 1624 MB (4 MB – 4096 MB)
Boot Order: ✓ Optical ✓ Hard Disk ✓ Floppy ✓ Network
Chipset: PIIX3
Pointing Device: USB Multi-Touch Tablet
Extended Features: ✓ Enable I/O APIC
☐ Enable EFI (special OSes only)
✓ Hardware Clock in UTC Time

General System Display Storage Audio Network Ports Shared Folders User Interface
Processor Acceleration
Tools
ubunutu_20.04_VM Powered Off
Kali-vg_default_1644 Powered Off
ubuntu-vg_default_1 Aborted
Cancel OK

Fig. 1.4 RAM setup


ubunutu_20.04_VM - Storage

Storage Devices:
Controller: IDE
Empty
Controller: SATA
ubunutu_20.04_VM.vdi

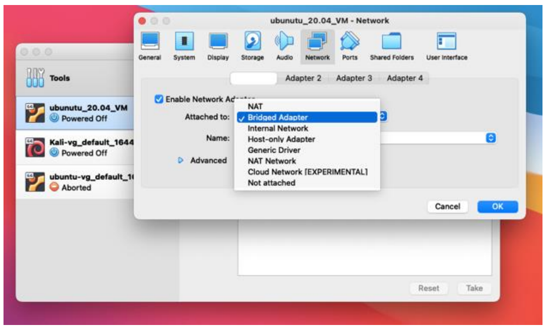
Attributes
Hard Disk: SATA Port 0
☐ Solid-state Drive
☐ Hot-pluggable

Information
Type (Format): Normal (VDI)
Virtual Size: 10.00 GB
Actual Size: 8.33 GB
Details: Dynamically allocated storage
Location: /Users/vinitsinha/VirtualBox ...
Attached to: ubunutu_20.04_VM
Encrypted with key: --

Lists all storage controllers for this machine and the virtual images and host drives attached to them.

General System Display Storage Audio Network Ports Shared Folders User Interface
Cancel OK

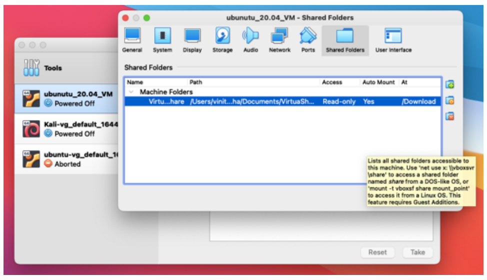
Fig. 1.5 Storage device setup


ubunutu_20.04_VM - Network

✓ Enable Network Adapter
Attached to: ✓ Bridged Adapter
NAT
Bridged Adapter
Internal Network
Host-only Adapter
Generic Driver
NAT Network
Cloud Network [EXPERIMENTAL]
Not attached

General System Display Storage Audio Network Ports Shared Folders User Interface
Adapter 2 Adapter 3 Adapter 4
Name:
Advanced
Cancel OK
Reset Take

Fig. 1.6 Network setup


ubunutu_20.04_VM - Shared Folders

Name	Path	Access	Auto Mount	At
Machine Folders				
Virtu...hare	/Users/vinit...ha/Documents/VirtuaSh...	Read-only	Yes	/Download

Lists all shared folders accessible to this machine. Use 'net use x: \\vboxsvr\share' to access a shared folder named *share* from a DOS-like OS, or 'mount -t vboxsf share mount_point' to access it from a Linux OS. This feature requires Guest Additions.

General System Display Storage Audio Network Ports Shared Folders User Interface
Reset Take

Fig. 1.7 Shared Folder setup

VII. KVM IMPLEMENTATION

7.1. Starting QEMU with KVM support –

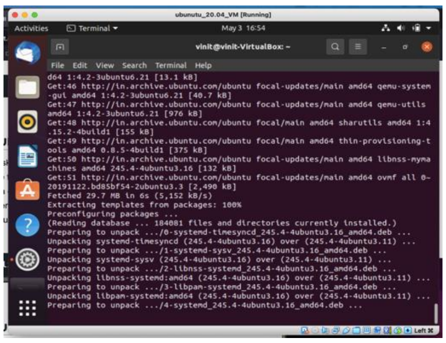
QEMU is basic and main element of KVM / QEMU in virtual environment. It provides support for H/W virtualization and emulator for processor establishment without taking support from kernel (Linux). Qemu executes itself in user space. For fast emulation of system in virtualization it takes support from drivers for handling hardware as well as software. Qemu operates in two operating modes as

1. Emulation in user mode – Where Qemu execute process that are compiled on different CPU core on different platform.

2. Emulation in full system mode – Where Qemu executes entire system process, which includes peripherals and entire CPU system.

Installation of Qemu-KVM in ubuntu on virtual box is shown in following **Fig. 5.8** using command as,

# apt-get install -y qemu



```
d64 1:4.2-3ubuntu6.21 [13.1 kB]
Get:46 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 qemu-system
-gui amd64 1:4.2-3ubuntu6.21 [40.7 kB]
Get:47 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 qemu-utils
amd64 1:4.2-3ubuntu6.21 [976 kB]
Get:48 http://in.archive.ubuntu.com/ubuntu focal/main amd64 sharutils amd64 1:4
.15.2-4build1 [155 kB]
Get:49 http://in.archive.ubuntu.com/ubuntu focal/main amd64 thin-provisioning-t
ools amd64 0.8.5-4build1 [375 kB]
Get:50 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnss-myma
chines amd64 245.4-4ubuntu3.16 [132 kB]
Get:51 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 ovmf all 0~
20191122.bd85bf54-2ubuntu3.3 [2,490 kB]
Fetched 29.7 MB in 6s (5,152 kB/s)
Extracting templates from packages: 100%
Preconfiguring packages ...
(Reading database ... 184081 files and directories currently installed.)
Preparing to unpack .../0-systemd-timesyncd_245.4-4ubuntu3.16_amd64.deb ...
Unpacking systemd-timesyncd (245.4-4ubuntu3.16) over (245.4-4ubuntu3.11) ...
Preparing to unpack .../1-systemd-sysv_245.4-4ubuntu3.16_amd64.deb ...
Unpacking systemd-sysv (245.4-4ubuntu3.16) over (245.4-4ubuntu3.11) ...
Preparing to unpack .../2-libnss-systemd_245.4-4ubuntu3.16_amd64.deb ...
Unpacking libnss-systemd:amd64 (245.4-4ubuntu3.16) over (245.4-4ubuntu3.11) ...
Preparing to unpack .../3-libpam-systemd_245.4-4ubuntu3.16_amd64.deb ...
Unpacking libpam-systemd:amd64 (245.4-4ubuntu3.16) over (245.4-4ubuntu3.11) ...
Preparing to unpack .../4-systemd_245.4-4ubuntu3.16_amd64.deb ...
```

ubunutu_20.04_VM [Running]
Activities
Terminal
May 3 16:54
vinit@vinit-VirtualBox: ~
File Edit View Search Terminal Help

Fig. 1.8 Qemu installation on Ubuntu

7.2. Lib-virt implementation –

In previous section Qemu was discussed. It is convenient to use Qemu command for initialization of virtual instances. But they do not provides configuration and control over life cycle of VM. To overcome this libvirt toolset is more useful, as it is used for configure, build, start and stop of VM as well as other operation like migration, termination of VM at any level. Libvirt support different virtualization technology as Xen, virtual box, Containers (LXC) and Qemu / KVM. Following **Fig. 5.9** shows libvirt implementation on ubuntu in virtual box.

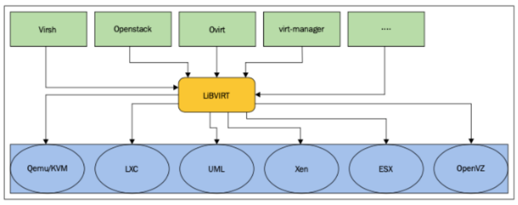
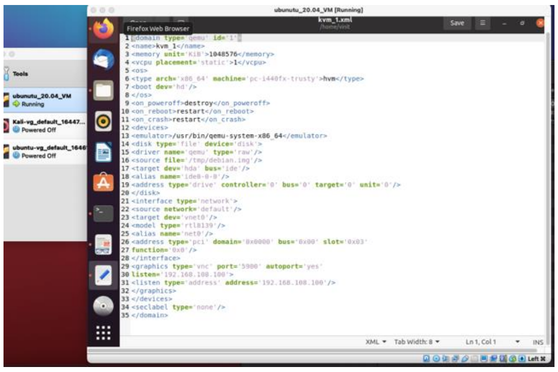


Fig. 1.9 Implementation of lib-virt

As shown in above, libvirt is consider as opensource API. It is also utilized for managing different hypervisor. It is treated as extra controlling layer in virtualization. Above figure shows different wings of libvirt, where it starts from virsh , it is command line interface for libvirt. It is again managed by high level management tool called as oVirt. The primary objective of libvirt is to provide stable and generalized platform to managed VMs running on different hypervisor.

7.3. Defining KVM instances –

For operation of virtual machine execution, instance generation is basic step towards VM handling. In this section virtual instance is created using simple XML file (configuration) as shown in following **fig. 1.10.**

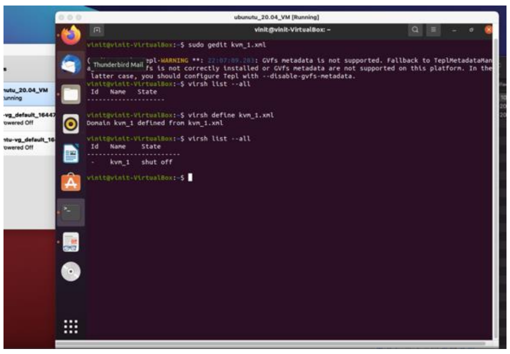


```
<domain type='qemu' id='1'>
<name>kvm_1</name>
<memory unit='KiB'>1048576</memory>
<vcpu placement='static'>1</vcpu>
<os>
<type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
<boot dev='hd'/>
</os>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>restart</on_crash>
<devices>
<emulator>/usr/bin/qemu-system-x86_64</emulator>
<disk type='file' device='disk'>
<driver name='qemu' type='raw'/>
<source file='/tmp/debian.img'/>
<target dev='hda' bus='ide'/>
<alias name='ide0-0-0'/>
<address type='drive' controller='0' bus='0' target='0' unit='0'/>
</disk>
<interface type='network'>
<source network='default'/>
<target dev='vnet0'/>
<model type='rtl8139'/>
<alias name='net0'/>
<address type='pci' domain='0x0000' bus='0x00' slot='0x03'
function='0x0'/>
</interface>
<graphics type='vnc' port='5900' autoport='yes'
listen='192.168.100.100'>
<listen type='address' address='192.168.100.100'/>
</graphics>
</devices>
<seclabel type='none'/>
</domain>
```

Fig. 1.10 Creation of KVM instance

Virtual instance can also be created using ,

# virsh define KVM_1.xml as shown in following **fig. 1.11**



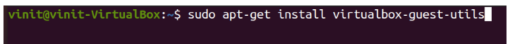
```
vinit@vinit-VirtualBox:~$ sudo gedit kvm_1.xml

(                epl-WARNING **: 22:07:09.283: GVfs metadata is not supported. Fallback to TeplMetadataMan
a_                fs is not correctly installed or GVfs metadata are not supported on this platform. In the
 latter case, you should configure Tepl with --disable-gvfs-metadata.
vinit@vinit-VirtualBox:~$ virsh list --all
 Id   Name   State
--------------------

vinit@vinit-VirtualBox:~$ virsh define kvm_1.xml
Domain kvm_1 defined from kvm_1.xml

vinit@vinit-VirtualBox:~$ virsh list --all
 Id   Name    State
------------------------
 -    kvm_1   shut off

vinit@vinit-VirtualBox:~$ 
```

Fig. 1.11 Defining KVM instance

7.4. Sharing directories between VM and Host OS –

While working with virtual box for managing multiple guest system, at crucial situation requirement of sharing of directory between Host and Guest system is at peak level. So to add on this feature, *virtual box guest addition* should be install on guest system. In this research MacOS / Windows 10 (any one) is taken as host system and ubuntu 22.04 is as Guest virtual machine.

For Ubuntu as guest OS, following command is used to guest addition , shown in following **fig. 5.12.**



```
vinit@vinit-VirtualBox:~$ sudo apt-get install virtualbox-guest-utils
```

Fig. 1.12 – Guest addition in ubuntu VM

Next step is to set share folder and session creation on Virtual box for sharing files from host to guest and vice-versa. Following figures shows steps for directory sharing procedure.

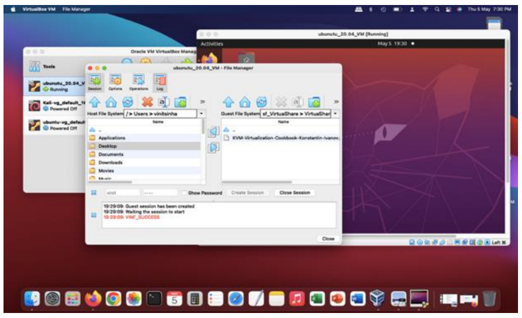
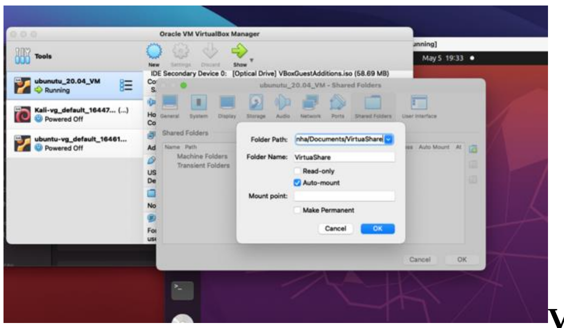


Fig. 1.13 File manager for host and guest



IDE Secondary Device 0: [Optical Drive] VBoxGuestAdditions.iso (58.69 MB)

Folder Path: nha/Documents/VirtuaShare
Folder Name: VirtuaShare
Read-only
[✓ Auto-mount]
Mount point:
Make Permanent

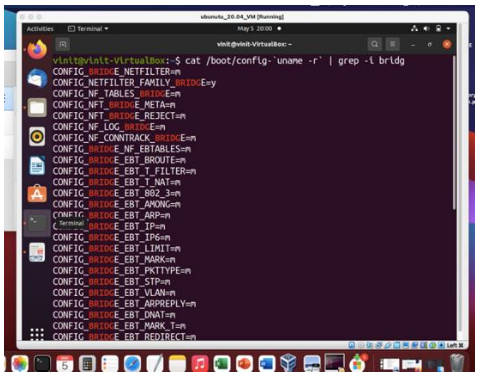
Fig. 1.14 – Directory sharing options in virtual box

7.5. KVM networking with libvirt –

By using libvirt different types of network can be define for KVM guest. For this XML definition, virsh and userspace tool (virt-install) is used effectively. To make network connection for KVM instances and host system some elements are need to configured which are as follows –

1. Linux bridge
2. Openvswitch (OVS)
3. Kernel modules
4. Userspace tool

All this software bridge technology responsible for making SDN (Software Defined Network) at various complex level. Here OVS and linux bridge both act as bridge network to connect all virtual interface of KVM. Following **fig. 5.15** shows command used to check kernel enable functionality for 802.1dx ethernet standard for bridge connection.



```
vinit@vinit-VirtualBox:~$ cat /boot/config-`uname -r` | grep -i bridg
CONFIG_BRIDGE_NETFILTER=m
CONFIG_NETFILTER_FAMILY_BRIDGE=y
CONFIG_NF_TABLES_BRIDGE=m
CONFIG_NFT_BRIDGE_META=m
CONFIG_NFT_BRIDGE_REJECT=m
CONFIG_NF_LOG_BRIDGE=m
CONFIG_NF_CONNTRACK_BRIDGE=m
CONFIG_BRIDGE_NF_EBTABLES=m
CONFIG_BRIDGE_EBT_BROUTE=m
CONFIG_BRIDGE_EBT_T_FILTER=m
CONFIG_BRIDGE_EBT_T_NAT=m
CONFIG_BRIDGE_EBT_802_3=m
CONFIG_BRIDGE_EBT_AMONG=m
CONFIG_BRIDGE_EBT_ARP=m
CONFIG_BRIDGE_EBT_IP=m
CONFIG_BRIDGE_EBT_IP6=m
CONFIG_BRIDGE_EBT_LIMIT=m
CONFIG_BRIDGE_EBT_MARK=m
CONFIG_BRIDGE_EBT_PKTTYPE=m
CONFIG_BRIDGE_EBT_STP=m
CONFIG_BRIDGE_EBT_VLAN=m
CONFIG_BRIDGE_EBT_ARPREPLY=m
CONFIG_BRIDGE_EBT_DNAT=m
CONFIG_BRIDGE_EBT_MARK_T=m
CONFIG_BRIDGE_EBT_REDIRECT=m
```

Fig. 1.15 Linux kernel bridge check

For manipulation of bridge network in ubuntu guest os, linux bridge package must be install , for that following command is used.

# sudo apt-get install bridge-utils

VIII. IMPLEMENTATION

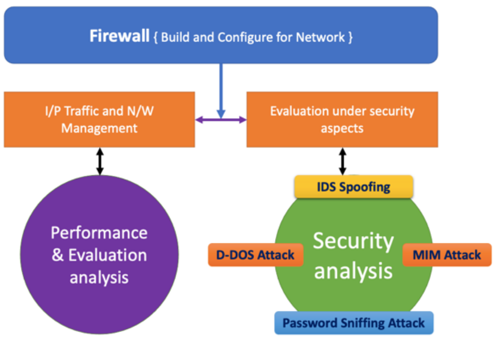

Firewall { Build and Configure for Network }
I/P Traffic and N/W Management
Evaluation under security aspects
Performance & Evaluation analysis
IDS Spoofing
D-DOS Attack
Security analysis
MIM Attack
Password Sniffing Attack

Fig. 1.16 Firewall Evaluation

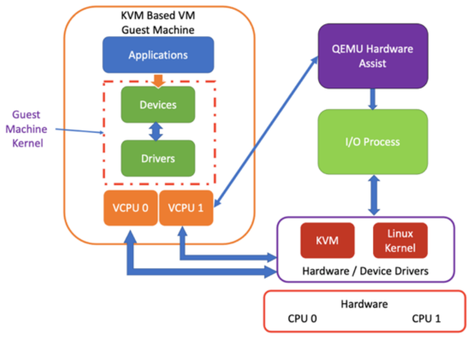
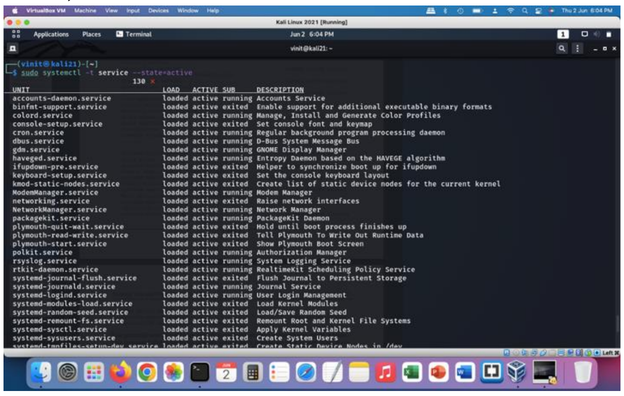


Fig. 1.17 KVM Evaluation

While managing security services on both Host and Guest system, their maintenance regarding fulfilment of security is required at top priority.

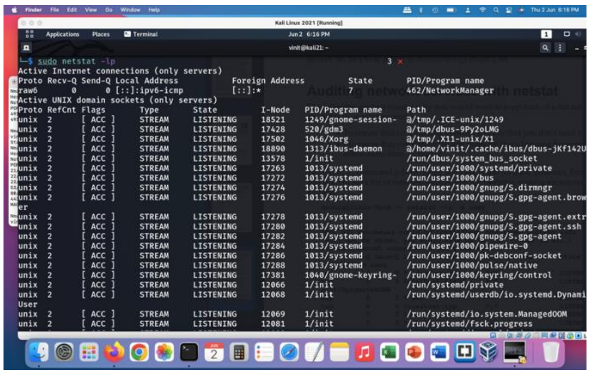
1. In linux (Kali), *systemctl* command is widely used for maintaining services with respect to network and server system. Following **fig. 1.18** shows system service status on guest system (Kali Linux) as,



```
┌──(vinit㉿kali21)-[~]
└─$ sudo systemctl -t service --state=active
                                130 ✗
 UNIT                              LOAD   ACTIVE SUB     DESCRIPTION
 accounts-daemon.service           loaded active running Accounts Service
 binfmt-support.service            loaded active exited  Enable support for additional executable binary formats
 colord.service                    loaded active running Manage, Install and Generate Color Profiles
 console-setup.service             loaded active exited  Set console font and keymap
 cron.service                      loaded active running Regular background program processing daemon
 dbus.service                      loaded active running D-Bus System Message Bus
 gdm.service                       loaded active running GNOME Display Manager
 haveged.service                   loaded active running Entropy Daemon based on the HAVEGE algorithm
 ifupdown-pre.service              loaded active exited  Helper to synchronize boot up for ifupdown
 keyboard-setup.service            loaded active exited  Set the console keyboard layout
 kmod-static-nodes.service         loaded active exited  Create list of static device nodes for the current kernel
 ModemManager.service              loaded active running Modem Manager
 networking.service                loaded active exited  Raise network interfaces
 NetworkManager.service            loaded active running Network Manager
 packagekit.service                loaded active running PackageKit Daemon
 plymouth-quit-wait.service        loaded active exited  Hold until boot process finishes up
 plymouth-read-write.service       loaded active exited  Tell Plymouth To Write Out Runtime Data
 plymouth-start.service            loaded active exited  Show Plymouth Boot Screen
 polkit.service                    loaded active running Authorization Manager
 rsyslog.service                   loaded active running System Logging Service
 rtkit-daemon.service              loaded active running RealtimeKit Scheduling Policy Service
 systemd-journal-flush.service     loaded active exited  Flush Journal to Persistent Storage
 systemd-journald.service          loaded active running Journal Service
 systemd-logind.service            loaded active running User Login Management
 systemd-modules-load.service      loaded active exited  Load Kernel Modules
 systemd-random-seed.service       loaded active exited  Load/Save Random Seed
 systemd-remount-fs.service        loaded active exited  Remount Root and Kernel File Systems
 systemd-sysctl.service            loaded active exited  Apply Kernel Variables
 systemd-sysusers.service          loaded active exited  Create System Users
 systemd-tmpfiles-setup-dev.service loaded active exited  Create Static Device Nodes in /dev
```

Fig. 1.18 System services on Kali Linux

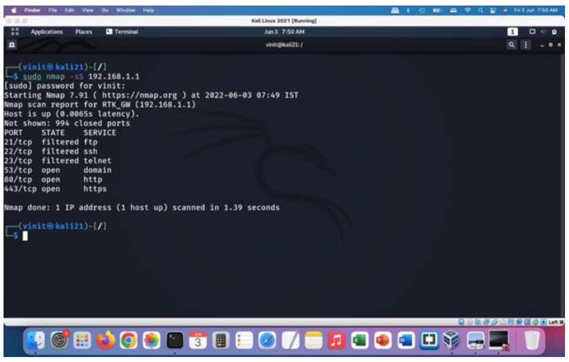
2. Like system services, network resources can also be maintained using *netstat* command, whose details output is shown in following **fig. 1.19**. Netstat provides track of network services on guest system, which insure about no unwanted services running on system like (malware – listening for network connection) and unwanted network services which consume resources.



```
└─$ sudo netstat -lp                                                    3 ✗
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
raw6       0      0 [::]:ipv6-icmp          [::]:*                  7           462/NetworkManager
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ACC ]     STREAM     LISTENING     18521    1249/gnome-session-  @/tmp/.ICE-unix/1249
unix  2      [ ACC ]     STREAM     LISTENING     17428    520/gdm3             @/tmp/dbus-9Py2oLMG
unix  2      [ ACC ]     STREAM     LISTENING     17502    1046/Xorg            @/tmp/.X11-unix/X1
unix  2      [ ACC ]     STREAM     LISTENING     18890    1313/ibus-daemon     @/home/vinit/.cache/ibus/dbus-jKf142U
unix  2      [ ACC ]     STREAM     LISTENING     13578    1/init               /run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     17263    1013/systemd         /run/user/1000/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     17272    1013/systemd         /run/user/1000/bus
unix  2      [ ACC ]     STREAM     LISTENING     17274    1013/systemd         /run/user/1000/gnupg/S.dirmngr
unix  2      [ ACC ]     STREAM     LISTENING     17276    1013/systemd         /run/user/1000/gnupg/S.gpg-agent.brow
er
unix  2      [ ACC ]     STREAM     LISTENING     17278    1013/systemd         /run/user/1000/gnupg/S.gpg-agent.extr
unix  2      [ ACC ]     STREAM     LISTENING     17280    1013/systemd         /run/user/1000/gnupg/S.gpg-agent.ssh
unix  2      [ ACC ]     STREAM     LISTENING     17282    1013/systemd         /run/user/1000/gnupg/S.gpg-agent
unix  2      [ ACC ]     STREAM     LISTENING     17284    1013/systemd         /run/user/1000/pipewire-0
unix  2      [ ACC ]     STREAM     LISTENING     17286    1013/systemd         /run/user/1000/pk-debconf-socket
unix  2      [ ACC ]     STREAM     LISTENING     17288    1013/systemd         /run/user/1000/pulse/native
unix  2      [ ACC ]     STREAM     LISTENING     17381    1040/gnome-keyring-  /run/user/1000/keyring/control
unix  2      [ ACC ]     STREAM     LISTENING     12066    1/init               /run/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     12068    1/init               /run/systemd/userdb/io.systemd.Dynami
User
unix  2      [ ACC ]     STREAM     LISTENING     12069    1/init               /run/systemd/io.system.ManagedOOM
unix  2      [ ACC ]     STREAM     LISTENING     12081    1/init               /run/systemd/fsck.progress
```

Fig. 1.19 Network services on guest system

3. Auditing network services with NMAP had more advantages, specially used in virtual environment like Kali linux as it support auditing services without log on to each module. NMAP help to scan network for identification of malware or intruder activity. Following **fig. 1.20** shows scanning of single machine on terminal using NMAP tool.



```
┌──(vinit㉿kali21)-[/]
└─$ sudo nmap -sS 192.168.1.1
[sudo] password for vinit:
Starting Nmap 7.91 ( https://nmap.org ) at 2022-06-03 07:49 IST
Nmap scan report for RTK_GW (192.168.1.1)
Host is up (0.0065s latency).
Not shown: 994 closed ports
PORT    STATE    SERVICE
21/tcp  filtered ftp
22/tcp  filtered ssh
23/tcp  filtered telnet
53/tcp  open     domain
80/tcp  open     http
443/tcp open     https

Nmap done: 1 IP address (1 host up) scanned in 1.39 seconds

┌──(vinit㉿kali21)-[/]
└─$ 
```

Fig. 1.20 Network services using NMAP

After executing NMAP on guest OS it shows state of machine ports as,

- **filtered** – port is blocked by guest firewall.
- **Open** – Port is not blocked by any firewall and services running on that port are on.
- **Closed** – Port is not block by firewall but services on port in not running.

IX. RESULT AND EVALUATION

Performance evaluation of KVM on basis of CPU, Memory, Network, Process and GPU is shown in following **table 1.1** followed by figures **(1.21, 1.22, 1.23)** in graph pattern.

		Parameters					
	Guest OS	Memory (MB)	Network (kbps)	CPU (%)	Process (%)	GPU (%)	Speed (Hz)
1	KALI 2021	67.6	492	12	9.5	9	2.74
2	Ubuntu 22.04	71.1	578	9	7	12	3.03

Table 1.1 KVM performance

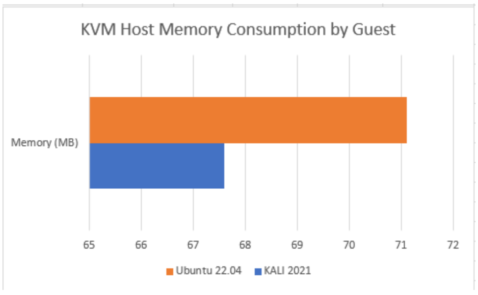


Fig. 1.21Memory Consumption Graph

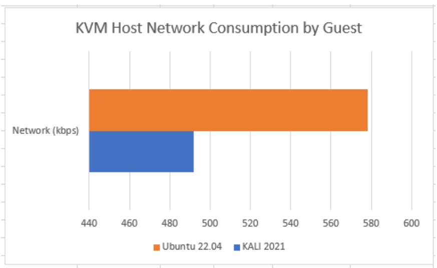


Fig. 1.22 Network Consumption Graph

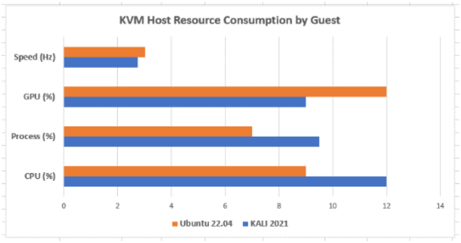


Fig. 1.23 Resource Consumption Graph

X. CONCLUSION

This research provides network security solutions to enhance the server security specially in Linux system environment. As per discussed in section 7 it generates auditing of system services followed by log analysis. Which helps to monitor security progress and track of problem-solving techniques. By running different high resource consume application, this research provides great resource support on heavy network traffic by using VB as a type II hypervisor technique.

As a contribution, this research provides detail of accuracy factors as type of hypervisor, virtualization methods and deployment of applications. While maintaining server security this research provides service support as,

1. Virtualization platform.
2. Shell scripting support for network security.
3. Firewall rules set up and configuration.
4. Server system scanning which related to network.
5. Comparison of security techniques for effective network security on virtual environment.

XI. FUTURE WORK

While developing security techniques on basis of virtualisation in Kali Linux as well as Ubuntu, some pitfalls faces in research which can be recovered in future with the following consideration as,

1. While running guest system on host an advanced firewall tool can be improved by setting it up on router and providing LAN base UI to administrator.
2. VPN tools are safe to use only win configured by targeting specific system.
3. Firewall configuration and installation will be in kickstart format through DHCP server on host system and it should reflect on guest system too.

REFERENCES

[1] Ganji, H. R., & Aghakhani, K. (2018). Provides a New Way to Enhance Security in the Linux Operating System. Emerging Science Journal, 2(5), 295.

[2] Patil, R., Dudeja, H., & Modi, C. (2019). Designing an efficient security framework for detecting intrusions in virtual network of cloud computing. Computers & Security, 85, 402–422.

[3] Li, Y., Huang, G., Wang, C., & Li, Y. (2019). Analysis framework of network security situational awareness and comparison of implementation methods. EURASIP Journal on Wireless Communications and Networking, 2019(1), 205.

[4] De Bock, Y., Mercelis, S., Broeckhove, J., & Hellinckx, P. (2020). Real-time virtualization with Xvisor. Internet of Things, 11, 100238.

[5] Compastié, M., Badonnel, R., Festor, O., & He, R. (2020). From virtualization security issues to cloud protection opportunities: An in-depth analysis of

system virtualization models. Computers & Security, 97, 101905.

[6] Yan, T., Liu, J., Niu, Q., Chen, J., Xu, S., & Niu, M. (2020). Network security protection technology for a cloud energy storage network controller. Global Energy Interconnection, 3(1), 85–97.

[7] Potdar, A. M., D G, N., Kengond, S., & Mulla, M. M. (2020). Performance Evaluation of Docker Container and Virtual Machine. Procedia Computer Science, 171, 1419–1428.

[8] Bahn, H., & Kim, J. (2020). Separation of Virtual Machine I/O in Cloud Systems. IEEE Access, 8, 223756– 223764.