

CS790 - Privacy Enhancing Technologies

Assignment 1 - Traffic Filtering

Shantanu Welling

February 2025

Contents

Task 1	1
Part A	1
Part B	4
Part C	5
Part D	6
Task 2	7
References	11

Task 1

Part A

Used FreeBSD for all 3 VMs.

1. To assign private IPs within the same subnet, I first added 3 network adapters for VM2 in its VirtualBox settings - 1 NAT which connects to the host for ssh (VM port 22 forwarded to host port 2222), 2 network adapters attached to 2 different internal networks (“`intnet1`” for VM1 subnet & “`intnet2`” for the VM3 subnet).
2. For VM2, NAT interface is named `em0`, `intnet1` interface - `em1` & `intnet2` interface - `em2`.
3. For VM2, `em0` is configured using DHCP by adding the line `ifconfig_em0="DHCP"` in `/etc/rc.conf`
4. Private IP address for `em1` is assigned statically by adding the line `ifconfig_em1='inet 192.168.10.2 netmask 255.255.255.0 up'` in `/etc/rc.conf`. This interface of VM2 is connected to VM1 and their subnet is given by `192.168.10.0/24`
5. Private IP address for `em2` is assigned statically by adding the line `ifconfig_em2='inet 192.168.20.2 netmask 255.255.255.0 up'` in `/etc/rc.conf`. This interface of VM2 is connected to VM3 and their subnet is given by `192.168.20.0/24`
6. These changes are reflected by running the command - `service netif restart && service routing restart`. The first command stops and then starts all network interfaces configured in `/etc/rc.conf` and the second command restarts routing services (like static routes, default gateways/routers) with the new routing settings in `/etc/rc.conf`

7. Similarly, I added 2 network adapters for VM1 in its VirtualBox settings - 1 NAT which connects to the host for ssh (VM port 22 forwarded to host port 2221) and a network adapter attached to the internal network ("intnet1" which is the subnet with VM2).
8. For VM1, NAT interface is named `em0`, `intnet1` interface - `em1`.
9. For VM1, `em0` is configured in the same way as VM2. Private IP address for `em1` is assigned statically by adding the line `ifconfig_em1='inet 192.168.10.3 netmask 255.255.255.0 up'` in `/etc/rc.conf`. This interface of VM1 is connected to VM2 interface `em1` and their subnet is given by `192.168.10.0/24`. Ran the 2 service restart commands to reflect the modifications, same as above.
10. Steps for VM3 are same as those for VM1, with the only difference being: internal network name `intnet2` whose interface is `em1` - which is connected to VM2 interface `em2`, IP address for VM3 `em1` is `192.168.20.3`.

```

VM1 VM2 VM3
root@vm1:~ # ifconfig
em0: flags=1008843<UP, BROADCAST, RUNNING, SIMPLEX, MULTICAST, LOWER_UP> metric 0 mtu 1500
    options=48505bb<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, JUMBO_MTU, VLAN_HWCSUM, TS04, LRO, VLAN_HWFILTER, VLAN_HWTSO, HWSTATS, MEXTPG>
    ether 08:00:27:72:1c:a4
    inet 10.0.2.15 netmask 0xffffffff broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe72:1ca4%em0 prefixlen 64 scopeid 0x1
    inet6 fd00::a00:27ff:fe72:1ca4 prefixlen 64 autoconf pltime 14400 vlttime 86400
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=23<PERFORMNUD, ACCEPT_RTADV, AUTO_LINKLOCAL>
em1: flags=1008843<UP, BROADCAST, RUNNING, SIMPLEX, MULTICAST, LOWER_UP> metric 0 mtu 1500
    options=48505bb<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, JUMBO_MTU, VLAN_HWCSUM, TS04, LRO, VLAN_HWFILTER, VLAN_HWTSO, HWSTATS, MEXTPG>
    ether 08:00:27:0a:b3:10
    inet 192.168.10.3 netmask 0xffffffff broadcast 192.168.10.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD, IFDISABLED, AUTO_LINKLOCAL>
lo0: flags=1008049<UP, LOOPBACK, RUNNING, MULTICAST, LOWER_UP> metric 0 mtu 16384
    options=680003<RXCSUM, TXCSUM, LINKSTATE, RXCSUM_IPV6, TXCSUM_IPV6>
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
    groups: lo
    nd6 options=21<PERFORMNUD, AUTO_LINKLOCAL>
root@vm1:~ #

```

VM1 configuration

```

VM1 VM2 VM3
root@vm2-fw:~ # ifconfig
em0: flags=1008843<UP, BROADCAST, RUNNING, SIMPLEX, MULTICAST, LOWER_UP> metric 0 mtu 1500
    options=48505bb<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, JUMBO_MTU, VLAN_HWCSUM, TS04, LRO, VLAN_HWFILTER, VLAN_HWTSO, HWSTATS, MEXTPG>
    ether 08:00:27:ea:14:f7
    inet 10.0.2.15 netmask 0xffffffff broadcast 10.0.2.255
    inet6 fe80::a00:27ff:feea:14f7%em0 prefixlen 64 scopeid 0x1
    inet6 fd00::a00:27ff:feea:14f7 prefixlen 64 autoconf pltime 14400 vlttime 86400
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=23<PERFORMNUD, ACCEPT_RTADV, AUTO_LINKLOCAL>
em1: flags=1008843<UP, BROADCAST, RUNNING, SIMPLEX, MULTICAST, LOWER_UP> metric 0 mtu 1500
    options=48505bb<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, JUMBO_MTU, VLAN_HWCSUM, TS04, LRO, VLAN_HWFILTER, VLAN_HWTSO, HWSTATS, MEXTPG>
    ether 08:00:27:d0:01:9e
    inet 192.168.10.2 netmask 0xffffffff broadcast 192.168.10.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD, IFDISABLED, AUTO_LINKLOCAL>
em2: flags=1008843<UP, BROADCAST, RUNNING, SIMPLEX, MULTICAST, LOWER_UP> metric 0 mtu 1500
    options=48505bb<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, JUMBO_MTU, VLAN_HWCSUM, TS04, LRO, VLAN_HWFILTER, VLAN_HWTSO, HWSTATS, MEXTPG>
    ether 08:00:27:ee:56:f5
    inet 192.168.20.2 netmask 0xffffffff broadcast 192.168.20.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD, IFDISABLED, AUTO_LINKLOCAL>
lo0: flags=1008049<UP, LOOPBACK, RUNNING, MULTICAST, LOWER_UP> metric 0 mtu 16384
    options=680003<RXCSUM, TXCSUM, LINKSTATE, RXCSUM_IPV6, TXCSUM_IPV6>
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x4
    groups: lo
    nd6 options=21<PERFORMNUD, AUTO_LINKLOCAL>
root@vm2-fw:~ #

```

VM2 configuration

```

VM1 VM2 VM3
root@vm3:~ # ifconfig
em0: flags=100843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST,LOWER_UP> metric 0 mtu 1500
    options=48505bb<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, JUMBO_MTU, VLAN_HWCSUM, TS04, LRO, VLAN_HWFILTER, VLAN_HWTSO, HWSTATS, MEXTPG>
    ether 08:00:27:78:a6:05
    inet 10.0.2.15 netmask 0xffffff00 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe78:a605%em0 prefixlen 64 scopeid 0x1
    inet6 fd00::a00:27ff:fe78:a605 prefixlen 64 autoconf pltime 14400 vlttime 86400
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=23<PERFORMNUD, ACCEPT_RTADV, AUTO_LINKLOCAL>
em1: flags=100843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST,LOWER_UP> metric 0 mtu 1500
    options=48505bb<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, JUMBO_MTU, VLAN_HWCSUM, TS04, LRO, VLAN_HWFILTER, VLAN_HWTSO, HWSTATS, MEXTPG>
    ether 08:00:27:9d:83:2a
    inet 192.168.20.3 netmask 0xffffff00 broadcast 192.168.20.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD, IFDISABLED, AUTO_LINKLOCAL>
lo0: flags=1008049<UP,LOOPBACK,RUNNING,MULTICAST,LOWER_UP> metric 0 mtu 16384
    options=680003<RXCSUM, TXCSUM, LINKSTATE, RXCSUM_IPV6, TXCSUM_IPV6>
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
    groups: lo
    nd6 options=21<PERFORMNUD, AUTO_LINKLOCAL>
root@vm3:~ #

```

VM3 configuration

```

root@vm1:~ # ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2): 56 data bytes
64 bytes from 192.168.10.2: icmp_seq=0 ttl=64 time=2.168 ms
64 bytes from 192.168.10.2: icmp_seq=1 ttl=64 time=4.448 ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=64 time=4.472 ms
^C
--- 192.168.10.2 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 2.168/3.696/4.472/1.080 ms
root@vm1:~ # ping 192.168.20.3
PING 192.168.20.3 (192.168.20.3): 56 data bytes
ping: sendto: No route to host
ping: sendto: No route to host
^C
--- 192.168.20.3 ping statistics ---
2 packets transmitted, 0 packets received, 100.0% packet loss

```

VM1 pings VM2, VM3

```

root@vm2-fw:~ # ping 192.168.10.3
PING 192.168.10.3 (192.168.10.3): 56 data bytes
64 bytes from 192.168.10.3: icmp_seq=0 ttl=64 time=1.259 ms
64 bytes from 192.168.10.3: icmp_seq=1 ttl=64 time=6.251 ms
^C
--- 192.168.10.3 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.259/3.755/6.251/2.496 ms
root@vm2-fw:~ # ping 192.168.20.3
PING 192.168.20.3 (192.168.20.3): 56 data bytes
64 bytes from 192.168.20.3: icmp_seq=0 ttl=64 time=2.409 ms
64 bytes from 192.168.20.3: icmp_seq=1 ttl=64 time=4.822 ms
^C
--- 192.168.20.3 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 2.409/3.615/4.822/1.206 ms

```

VM2 pings VM1, VM3

```

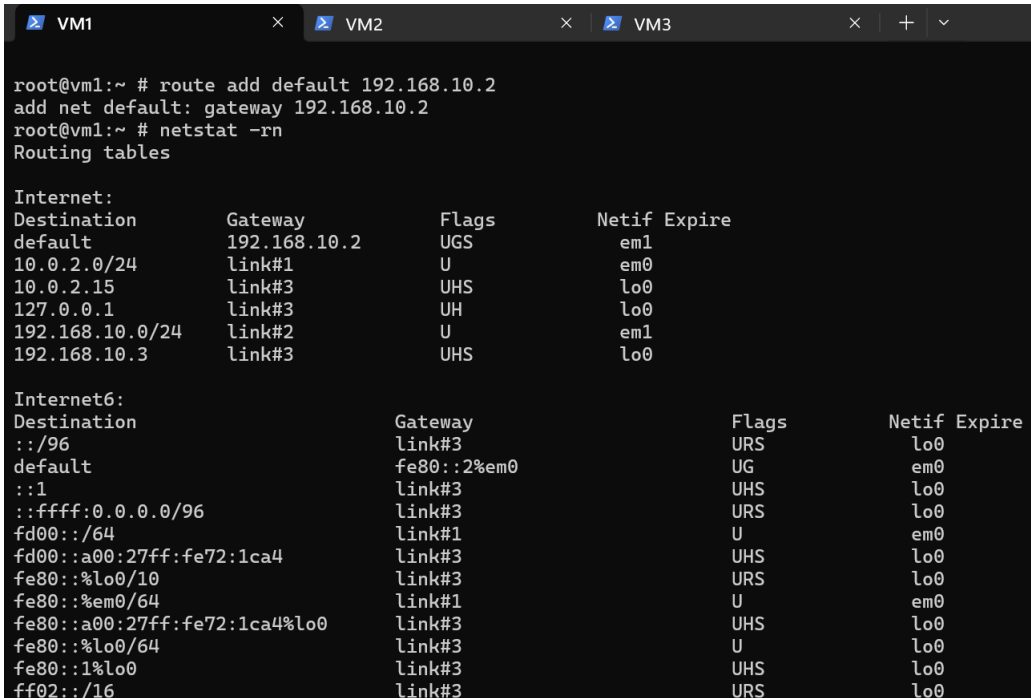
root@vm3:~ # ping 192.168.20.2
PING 192.168.20.2 (192.168.20.2): 56 data bytes
64 bytes from 192.168.20.2: icmp_seq=0 ttl=64 time=7.820 ms
64 bytes from 192.168.20.2: icmp_seq=1 ttl=64 time=4.582 ms
^C
--- 192.168.20.2 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 4.582/6.201/7.820/1.619 ms
root@vm3:~ # ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2): 56 data bytes
ping: sendto: No route to host
ping: sendto: No route to host
^C
--- 192.168.10.2 ping statistics ---
2 packets transmitted, 0 packets received, 100.0% packet loss

```

VM3 pings VM2, VM1

Part B

1. Enable IP forwarding on VM2 by adding the line `gateway_enable="YES"` to `/etc/rc.conf`. In order to reflect the change immediately without rebooting I used the command `sysctl net.inet.ip.forwarding=1`
2. Configured VM1's default route to VM2 by adding `defaultrouter="192.168.10.2"` to VM1's `/etc/rc.conf`. Equivalent command: `route add default 192.168.10.2`. Verify the change with the command `netstat -rn` which shows the system's routing table.
3. Similarly, configured VM3's default route to VM2 by adding `defaultrouter="192.168.10.2"` to VM1's `/etc/rc.conf`.
4. Pinged VM3 from VM1. Ran traceroute from VM1 to VM3. Results below.



```

root@vm1:~ # route add default 192.168.10.2
add net default: gateway 192.168.10.2
root@vm1:~ # netstat -rn
Routing tables

Internet:
Destination      Gateway          Flags           Netif Expire
default          192.168.10.2    UGS             em1
10.0.2.0/24      link#1          U               em0
10.0.2.15        link#3          UHS             lo0
127.0.0.1        link#3          UH              lo0
192.168.10.0/24  link#2          U               em1
192.168.10.3     link#3          UHS             lo0

Internet6:
Destination      Gateway          Flags           Netif Expire
::/96            link#3          URS             lo0
default          fe80::2%em0     UG              em0
::1              link#3          UHS             lo0
::ffff:0.0.0.0/96 link#3          URS             lo0
fd00::/64        link#1          U               em0
fd00::a00:27ff:fe72:1ca4 link#3          UHS             lo0
fe80::%lo0/10    link#3          URS             lo0
fe80::%em0/64    link#1          U               em0
fe80::a00:27ff:fe72:1ca4%lo0 link#3          UHS             lo0
fe80::%lo0/64    link#3          U               lo0
fe80::1%lo0      link#3          UHS             lo0
ff02::/16        link#3          URS             lo0

```

VM1 routing table

```

root@vm3:~ # netstat -rn
Routing tables

Internet:
Destination        Gateway             Flags               Netif Expire
default            192.168.20.2       UGS                 em1
10.0.2.0/24        link#1              U                   em0
10.0.2.15          link#3              UHS                 lo0
127.0.0.1          link#3              UH                  lo0
192.168.20.0/24    link#2              U                   em1
192.168.20.3       link#3              UHS                 lo0

Internet6:
Destination        Gateway             Flags               Netif Expire
::/96              link#3              URS                 lo0
::1                link#3              UHS                 lo0
::ffff:0.0.0.0/96  link#3              URS                 lo0
fd00::/64          link#1              U                   em0
fd00::a00:27ff:fe78:a605 link#3              UHS                 lo0
fe80::%lo0/10      link#3              URS                 lo0
fe80::%em0/64      link#1              U                   em0
fe80::a00:27ff:fe78:a605%lo0 link#3              UHS                 lo0
fe80::%lo0/64      link#3              U                   lo0
fe80::1%lo0        link#3              UHS                 lo0
ff02::/16          link#3              URS                 lo0

```

VM3 routing table

```

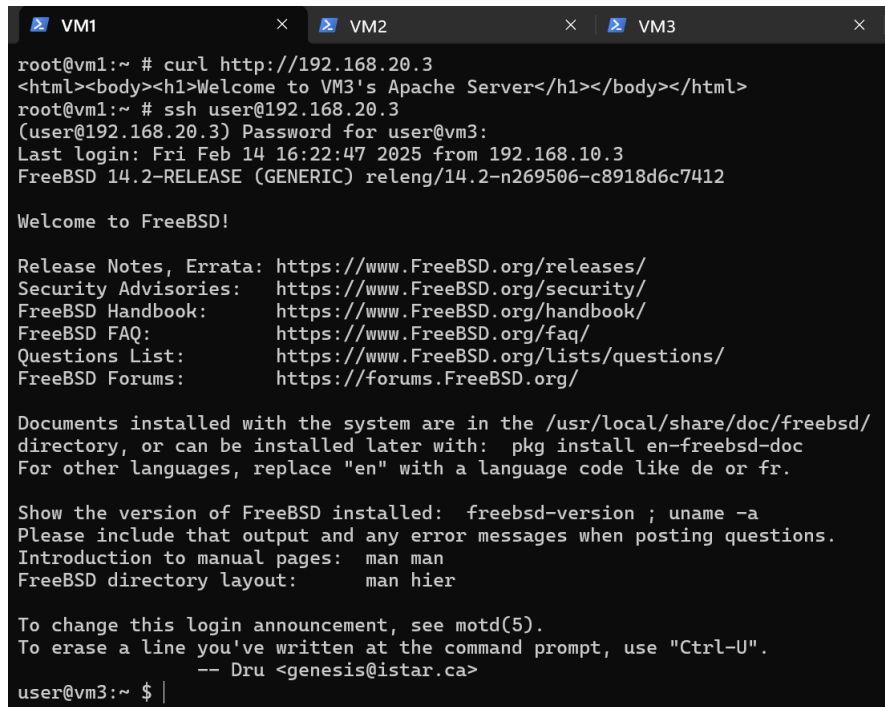
root@vm1:~ # ping 192.168.20.2
PING 192.168.20.2 (192.168.20.2): 56 data bytes
64 bytes from 192.168.20.2: icmp_seq=0 ttl=64 time=1.928 ms
64 bytes from 192.168.20.2: icmp_seq=1 ttl=64 time=3.985 ms
^C
--- 192.168.20.2 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.928/2.957/3.985/1.029 ms
root@vm1:~ # ping 192.168.20.3
PING 192.168.20.3 (192.168.20.3): 56 data bytes
64 bytes from 192.168.20.3: icmp_seq=0 ttl=63 time=3.759 ms
64 bytes from 192.168.20.3: icmp_seq=1 ttl=63 time=6.723 ms
^C
--- 192.168.20.3 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 3.759/5.241/6.723/1.482 ms
root@vm1:~ # traceroute 192.168.20.3
traceroute to 192.168.20.3 (192.168.20.3), 64 hops max, 40 byte packets
 1  192.168.10.2 (192.168.10.2)  1.938 ms  2.825 ms  2.324 ms
 2  192.168.20.3 (192.168.20.3)  6.658 ms  3.554 ms  4.269 ms

```

VM1 pings VM2:em2, VM3 and VM1 traceroute to VM3 which goes via VM2:em2

Part C

1. Installed apache24. Added `apache24.enable="YES"` to VM3's `/etc/rc.conf`. Equivalent command `sysrc apache24_enable=YES`.
2. `service apache24 start`. Verify server status - `service apache24 status`.
3. Set `ServerName` as `192.168.20.3` in `/usr/local/etc/apache24/httpd.conf` because Apache couldn't resolve the hostname of the system or determine the FQDN. Then ran `service apache24 restart`.
4. Created a web directory (`mkdir -p /usr/local/www/apache24/data`) and made simple `index.html` in the data folder.
5. Sent HTTP GET from VM1 to the webserver running on VM3 using the following command: `curl http://192.168.20.3` and got the `index.html` as response. I even hosted an SSH server on VM3 and I could establish ssh connection from VM1 to VM3 since VM2 firewall wasn't configured yet. Results below.



```

VM1 VM2 VM3
root@vm1:~ # curl http://192.168.20.3
<html><body><h1>Welcome to VM3's Apache Server</h1></body></html>
root@vm1:~ # ssh user@192.168.20.3
(user@192.168.20.3) Password for user@vm3:
Last login: Fri Feb 14 16:22:47 2025 from 192.168.10.3
FreeBSD 14.2-RELEASE (GENERIC) releng/14.2-n269506-c8918d6c7412

Welcome to FreeBSD!

Release Notes, Errata: https://www.FreeBSD.org/releases/
Security Advisories: https://www.FreeBSD.org/security/
FreeBSD Handbook: https://www.FreeBSD.org/handbook/
FreeBSD FAQ: https://www.FreeBSD.org/faq/
Questions List: https://www.FreeBSD.org/lists/questions/
FreeBSD Forums: https://forums.FreeBSD.org/

Documents installed with the system are in the /usr/local/share/doc/freebsd/
directory, or can be installed later with: pkg install en-freebsd-doc
For other languages, replace "en" with a language code like de or fr.

Show the version of FreeBSD installed: freebsd-version ; uname -a
Please include that output and any error messages when posting questions.
Introduction to manual pages: man man
FreeBSD directory layout: man hier

To change this login announcement, see motd(5).
To erase a line you've written at the command prompt, use "Ctrl-U".
-- Dru <genesis@istar.ca>
user@vm3:~ $ |

```

VM1 HTTP GET request to VM3's webserver and VM1 ssh to VM3

Part D

1. Added firewall rules in `/etc/pf.conf` file as shown below:

```

root@vm2-fw:~ # cat /etc/pf.conf
# Default policy: block all traffic unless specified otherwise
block all

# Allow HTTP traffic (port 80) to/fro VM3 (on em2)
pass in on em2 proto tcp to port 80
pass out on em2 proto tcp to port 80

# Allow all on em1
pass out on em1
pass in on em1

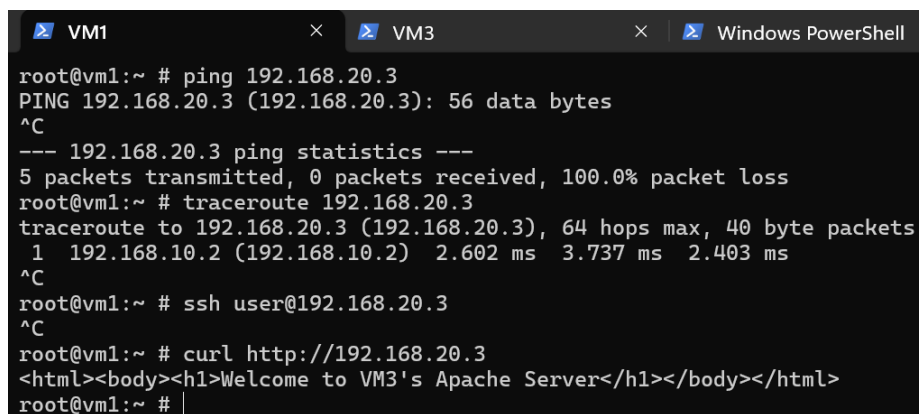
# Allow all on em0
pass out on em0
pass in on em0
root@vm2-fw:~ # pfctl -sr
block drop all
pass out on em2 proto tcp from any to any port = http flags S/SA keep state
pass out on em1 all flags S/SA keep state
pass out on em0 all flags S/SA keep state
pass in on em2 proto tcp from any to any port = http flags S/SA keep state
pass in on em1 all flags S/SA keep state
pass in on em0 all flags S/SA keep state
root@vm2-fw:~ #

```

VM2 Firewall Rules

2. PF reads rules from top to bottom; the last rule in a rule set that matches a packet or connection is the one that is applied. So, the first rule is to block all traffic by default unless specified by other rules which are specified later. All packets match the first rule but the behaviour could be overridden by subsequent rules. All the subsequent rules allow the packet to pass through based on the given rule conditions. If a packet doesn't match the pass-through latter rules, then it is blocked as per the first rule.

- Rules 4-7 allow all incoming and outgoing packets on interfaces `em0` and `em1`, since the firewall's functionality is to block all packets going to VM3 webserver (via interface `em2`) other than the ones with TCP protocol and port 80. So, rules 2 & 3 allow all incoming and outgoing packets on interface `em2` which have TCP protocol and port 80. Therefore, the VM2 firewall filters packets on the link connecting VM2:`em2` to VM3 because all the packets heading to VM3 would have to go through VM2:`em2` first.
- Added lines `pf_enable="yes"` and `pflog_enable="yes"` in VM2's `/etc/rc.conf` to enable packet filtering based on the rules given in `pf.conf`. Equivalent commands: `sysrc pf_enable=yes, sysrc pflog_enable=yes`.
- Started pf: `service pf start && service pflog start` or `pfctl -e`.
- Ran `pfctl -sr` to view the rules loaded in the firewall. Output shown in the image above.
- Finally, pinged VM3 from VM1 (blocked), traceroute from VM1 to VM3 (blocked), ssh from VM1 to VM3 (blocked). HTTP GET request from VM1 to VM3 webserver succeeds. See the image below.



```

VM1 VM3 Windows PowerShell
root@vm1:~ # ping 192.168.20.3
PING 192.168.20.3 (192.168.20.3): 56 data bytes
^C
--- 192.168.20.3 ping statistics ---
5 packets transmitted, 0 packets received, 100.0% packet loss
root@vm1:~ # traceroute 192.168.20.3
traceroute to 192.168.20.3 (192.168.20.3), 64 hops max, 40 byte packets
 1  192.168.10.2 (192.168.10.2)  2.602 ms  3.737 ms  2.403 ms
^C
root@vm1:~ # ssh user@192.168.20.3
^C
root@vm1:~ # curl http://192.168.20.3
<html><body><h1>Welcome to VM3's Apache Server</h1></body></html>
root@vm1:~ #

```

Task 2

Assumption: all incoming ICMP echo packets on VM2 are blocked irrespective of their interface.
Steps followed:

- Disable firewall and flush rules on VM2 by `pfctl -d && pfctl -F all`
- Run `make` (make `clean` if need to recompile) and load the kernel module using `kldload ./icmpfw.ko` on VM2
- ping 192.168.20.3 (VM3) or VM2: 192.168.20.2, 192.168.10.2 from VM1. It would be blocked and the dropped packets' size and number would be printed on VM2 terminal and could also be viewed through `dmesg` logs.
- Traceroute from VM1 to VM3 works normally but when I explicitly specify the protocol in the traceroute command using flag `-P ICMP` then it times out. From this I inferred that traceroute by default doesn't use ICMP echo.

- Another observation: ICMP traceroute packet size is 48 bytes whereas ping packet size is 84 bytes.
- Run `kldunload ./icmpfw.ko` to unload the kernel module. This prints the total number of packets dropped by the firewall (to both terminal and `dmesg` logs).
- The HTTP GET from VM1 to VM3 webserver still works as shown below because the firewall doesn't block those packets.

```

root@vm2-fw:~ # pfctl -d && pfctl -F all
pfctl: pf not enabled
root@vm2-fw:~ # make clean && make
rm -f export_syms machine x86 i386 icmpfw.ko icmpfw.kld icmpfw.o opt_global.h
machine -> /usr/src/sys/amd64/include
x86 -> /usr/src/sys/x86/include
i386 -> /usr/src/sys/i386/include
touch opt_global.h
Warning: Object directory not changed from original /root
cc -O2 -pipe -fno-strict-aliasing -Werror -D_KERNEL -DKLD_MODULE -nostdinc -include /root/opt_global.h -I. -I/usr/src/sys -I/usr/src/sys/contrib/ck/include -fno-common -fno-omit-frame-pointer -mno-omit-leaf-frame-pointer -fdebug-prefix-map=/machine=/usr/src/sys/amd64/include -fdebug-prefix-map=/x86=/usr/src/sys/x86/include -fdebug-prefix-map=/i386=/usr/src/sys/i386/include -MD -MF.de
pend.icmpfw.o -MTicmpfw.o -mmodel=kernel -mno-red-zone -mno-mmx -mno-sse -msoft-float -fno-asynchronous-unwind-tables -ffreestanding -fwrapv -fstack-protector -Wall -Wstrict-prototypes -Wmissing-prototypes -Wpointer-arith -Wcast-qual -Wundef -Wno-pointer-sign -D__
printf__=__freebsd_kprintf__ -Wmissing-include-dirs -fdiagnostics-show-option -Wno-unknown-pragmas -Wno-error=tautological-compare -W
no-error=empty-body -Wno-error=parentheses-equality -Wno-error=unused-function -Wno-error=pointer-sign -Wno-error=shift-negative-valu
e -Wno-address-of-packed-member -Wno-format-zero-length -mno-aes -mno-avx -std=gnu99 -c icmpfw.c -o icmpfw.o
ld -m elf_x86_64_fbsd -warn-common --build-id=sha1 -T /usr/src/sys/conf/ldscript.kmod.amd64 -r -o icmpfw.ko icmpfw.o
:~ export_syms
awk -f /usr/src/sys/conf/kmod_syms.awk icmpfw.ko export_syms | xargs -J% objcopy % icmpfw.ko
objcopy --strip-debug icmpfw.ko
root@vm2-fw:~ # kldload ./icmpfw.ko
root@vm2-fw:~ #

```

Setting up the kernel module on VM2

```

root@vm1:~ # ping 192.168.20.3
PING 192.168.20.3 (192.168.20.3): 56 data bytes
^C
--- 192.168.20.3 ping statistics ---
2 packets transmitted, 0 packets received, 100.0% packet loss
root@vm1:~ # traceroute 192.168.20.3
traceroute to 192.168.20.3 (192.168.20.3), 64 hops max, 40 byte packets
 1 192.168.10.2 (192.168.10.2) 1.822 ms 1.888 ms 1.810 ms
 2 192.168.20.3 (192.168.20.3) 10.706 ms 2.157 ms 3.060 ms
root@vm1:~ # traceroute -P ICMP 192.168.20.3
traceroute to 192.168.20.3 (192.168.20.3), 64 hops max, 48 byte packets
 1 * * *
 2 *^C
root@vm1:~ # curl http://192.168.20.3
<html><body><h1>Welcome to VM3's Apache Server</h1></body></html>
root@vm1:~ #
root@vm1:~ # # after unloading
root@vm1:~ # ping 192.168.20.3
PING 192.168.20.3 (192.168.20.3): 56 data bytes
64 bytes from 192.168.20.3: icmp_seq=0 ttl=63 time=3.418 ms
64 bytes from 192.168.20.3: icmp_seq=1 ttl=63 time=7.632 ms
^C
--- 192.168.20.3 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 3.418/5.525/7.632/2.107 ms

```

Test commands from VM1 timing out


```
root@vm2-fw:~ # ICMP Firewall Module Loaded

root@vm2-fw:~ # Dropped ICMP Echo Request: 84 bytes
Dropped ICMP Echo Request: 84 bytes

root@vm2-fw:~ # Dropped ICMP Echo Request: 48 bytes
Dropped ICMP Echo Request: 48 bytes
Dropped ICMP Echo Request: 48 bytes
Dropped ICMP Echo Request: 48 bytes
Dropped ICMP Echo Request: 48 bytes

root@vm2-fw:~ # kldunload ./icmpfw.ko
ICMP Firewall Module Unloaded. Total ICMP Packets Dropped: 7
root@vm2-fw:~ # █
```

VM2 firewall in action blocking VM1's ICMP Packets

Description of the kernel module:

The `load_module` function is responsible for registering and unregistering the packet filter hook. The function takes three parameters:

- **module** – A pointer to the kernel module structure.
- **event** – The action being performed (`MOD_LOAD` or `MOD_UNLOAD`).
- **arg** – Additional data (unused in this case).

Control flow:

1. If the event is `MOD_LOAD`, it registers the filtering function with the packet filter framework.
2. If the event is `MOD_UNLOAD`, it removes the filtering hook.
3. If the event is unrecognized, it returns `EOPNOTSUPP`.

The `pfil` framework allows for a specified function or a list of functions to be invoked for every incoming or outgoing packet for a particular network I/O stream.

1. While loading the module, the `pfil` hook is added using `pfil_add_hook` function which takes as argument `struct pfil_hook_args`.
2. This struct contains the module name and the pointer to the function (`pa_mbuf_chk` - packet message buffer check function) that processes each packet coming to the **Packet Filter Point (PFP)** which is given by the Head to which this hook is linked to. It also contains the packet type which the hook function would process (`PFIL_TYPE_IP4` in our case).
3. Linking of a hook to a particular PFP head is done by the function `pfil_link` which takes as argument `struct pfil_link_args`.
4. This struct stores:
 - (a) the `pfil_hook` which was added previously
 - (b) flags: `PFIL_IN`, `PFIL_HOOPTR` in our case to specify that only inbound packets need to be processed and the pointer to the hook is present within the struct, so no need to search the hook by its name
 - (c) PFP headname: `"inet"` in our case since we want to only process IPv4 packets. By default kernel creates the following heads:

- `"inet"` - IPv4 packets
- `"inet6"` - IPv6 packets
- `"ethernet"` - Link layer packets.

Default rulesets are automatically linked to these heads to preserve historical behaviour.

5. So basically, kernel has certain PFP heads for different types of packets. Each head has a list of hook functions, each hook function processes all the packets that come to that head and depending on the filtering done by the hook function, the packet is dropped, modified or allowed to pass through.
6. I created an ICMP filter function, added it as a hook and linked it to the `"inet"`-IPv4 PFP head.

My `icmp_filter` function is responsible for inspecting network packets and deciding whether to drop them.

Function parameters:

- `mp` – Pointer to the packet buffer (mbuf).
- `ifp` – Pointer to the network interface.
- `dir` – Packet direction (inbound or outbound).
- `arg` – Additional arguments (unused).
- `inp` – Pointer to the protocol control block (unused).

Control flow of the filter function:

1. Checks if the packet is valid.
2. Ensures it has at least an IP header.
3. Extracts the IP header and checks if the protocol is ICMP.
4. Ensures the packet is long enough to contain a complete ICMP header.
5. Extracts the ICMP header and checks if the type is `ICMP_ECHO` (ping request).
6. If so, the packet is freed and dropped.
7. Otherwise, the packet is allowed to pass.

Control Flow Between Functions:

1. The module is loaded via `kldload`, triggering `load_module(MOD_LOAD)`.
2. `load_module` registers the ICMP filter function `icmp_filter` with the FreeBSD packet filter.
3. When an inbound packet arrives, `icmp_filter` is invoked.
4. If the packet is an ICMP Echo Request, it is dropped.
5. When the module is unloaded via `kldunload`, `load_module(MOD_UNLOAD)` removes the filter.

References

1. [Pfil Error Debugging](#)
2. [Pfil Manpages](#)
3. [FreeBSD Source code](#)