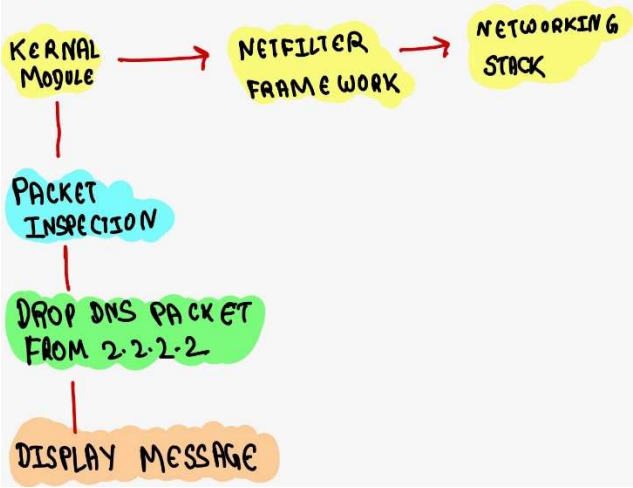
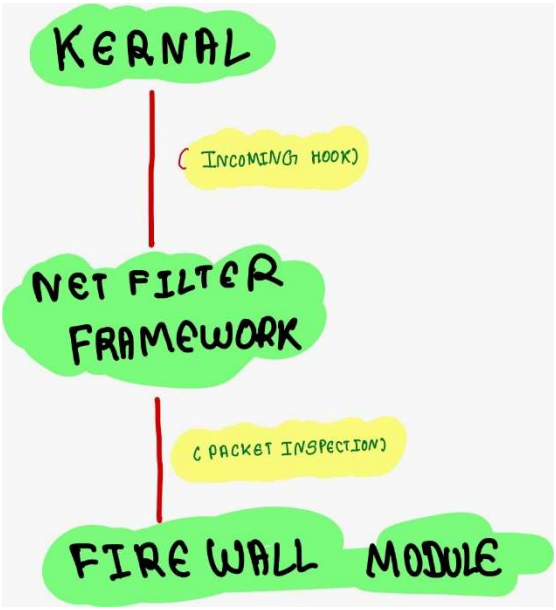
Kernel Module for Firewall to Block DNS Packets

Write a Kernel module program to implement a firewall which blocks DNS packets at port 53 coming from ip address 2.2.2.2. Show the message " Packet dropeed" when the packet calls the function at Incoming hook of Netfilter framework.



A kernel module is a piece of code that can be dynamically loaded and unloaded into the Linux kernel to extend its functionality. The Netfilter framework in Linux provides hooks into various points in the networking stack, allowing kernel modules to intercept and manipulate network packets.

Hooks are predefined points within the networking stack of an operating system where custom code can intercept and manipulate network packets.

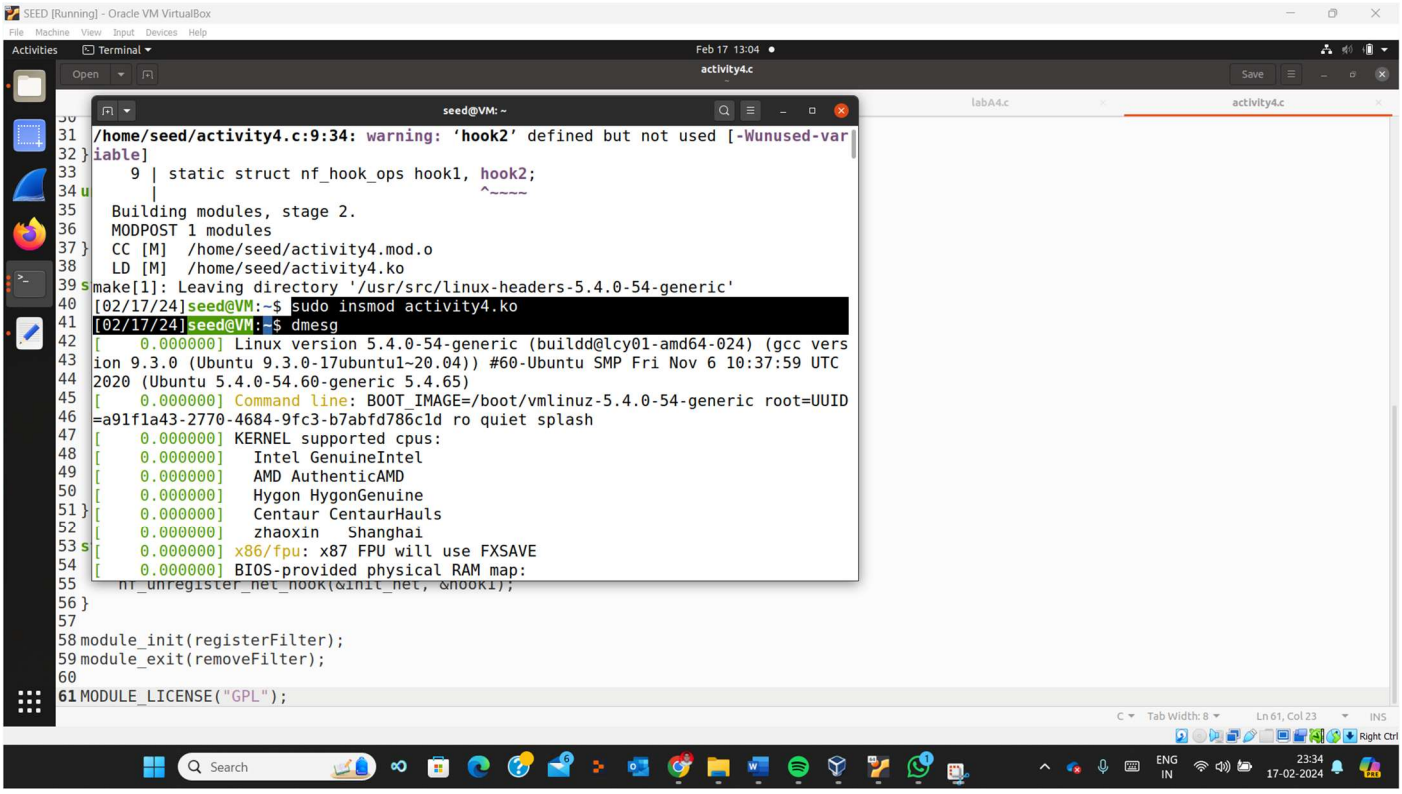


- The "Kernel" represents the Linux kernel.
- The "Netfilter Framework" is where the hooks are implemented.
- The "Firewall Module" is your kernel module that registers a function to be called at the Incoming hook.
- The arrows indicate the flow of network packets through the various components.
- At the Incoming hook, the firewall module intercepts the packets, inspects them, and takes action accordingly, such as dropping DNS packets from IP address 2.2.2.2 and displaying a message.

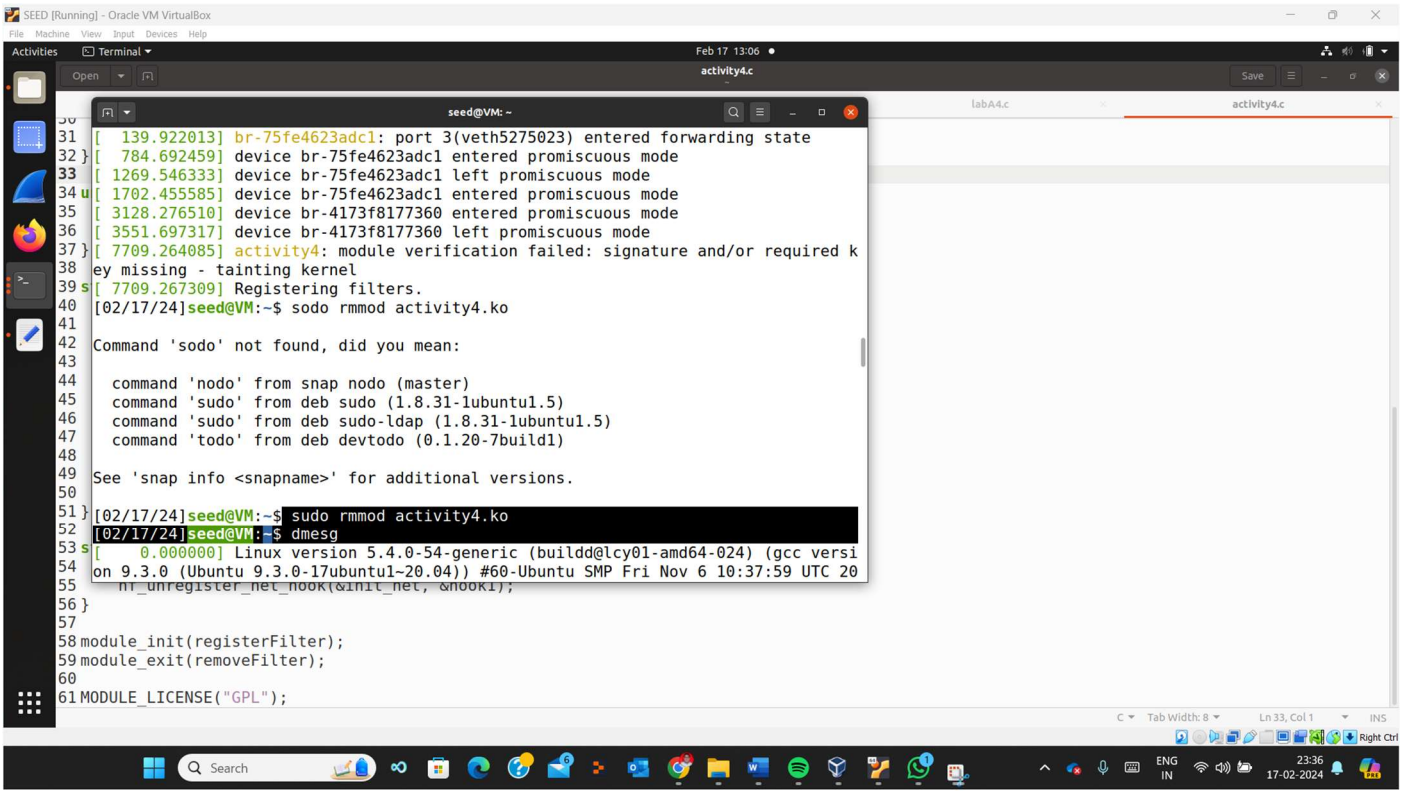
```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/netfilter.h>
#include <linux/skbuff.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/udp.h>

static struct nf_hook_ops hook1, hook2;

unsigned int hello1(void *priv, struct sk_buff *skb,
                    const struct nf_hook_state *state) {
   struct iphdr *ip_header = ip_hdr(skb);
   struct udphdr *udp_header;
   struct tcphdr *tcp_header;

   if (ip_header->protocol == IPPROTO_UDP) {
      udp_header = udp_hdr(skb);
      if (ntohs(udp_header->dest) == 53) {
         printk(KERN_INFO "*** Blocking UDP traffic on port 53\n");
         return NF_DROP;
      }
   } else if (ip_header->protocol == IPPROTO_TCP) {
      tcp_header = tcp_hdr(skb);
      if (ntohs(tcp_header->dest) == 53) {
         printk(KERN_INFO "*** Blocking TCP traffic on port 53\n");
```

```
            return NF_DROP;
        }
    }

    return NF_ACCEPT;
}

unsigned int hello2(void *priv, struct sk_buff *skb,
                    const struct nf_hook_state *state) {
    return NF_ACCEPT; // This hook will not be used for blocking port 53
}

static int __init registerFilter(void) {
    printk(KERN_INFO "Registering filters.\n");

    hook1.hook = hello1;
    hook1.hooknum = NF_INET_LOCAL_OUT;
    hook1.pf = PF_INET;
    hook1.priority = -100;
    nf_register_net_hook(&init_net, &hook1);

    // hook2 will not be used for blocking port 53, so it's not modified

    return 0;
}

static void __exit removeFilter(void) {
    printk(KERN_INFO "The filters are being removed.\n");
    nf_unregister_net_hook(&init_net, &hook1);
}

module_init(registerFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

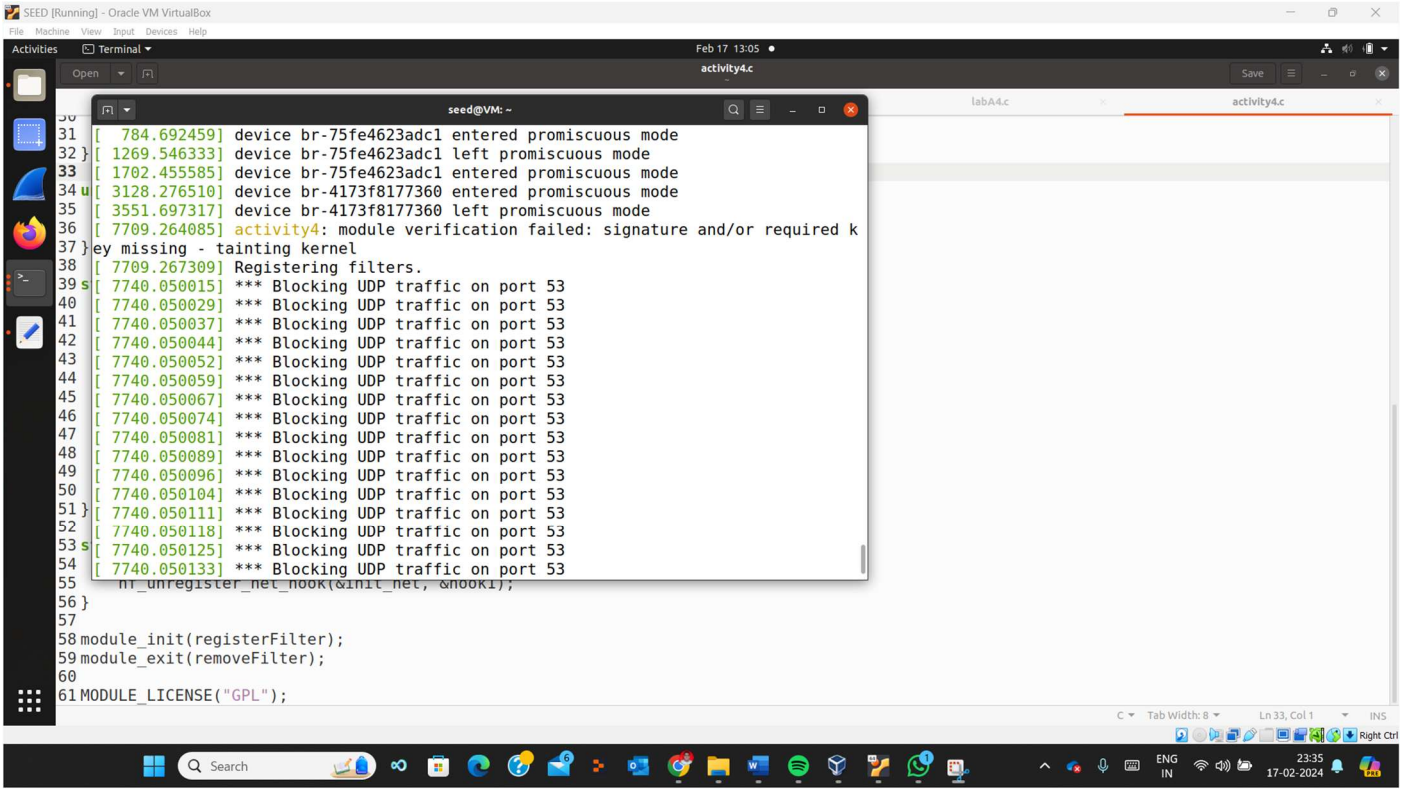
- Converting c file in .ko file by the help of makefile and inserting a module in linux kernal and dmesg for displaying the system message buffer, providing information about kernel and device activity.



- Using **rmmod** for removing module from the Linux kernel.



- Dmesg again to check that the module is gone or not



Learning

- Get to know about basics of kernal programming by which we can Develop code that directly interacts with the operating system core, gaining insights into system-level functionality.
- To understand about the role of Makefiles in the creation and management of kernel modules.
- Explore the creation and utilization of kernel object files (KO), enabling dynamic loading and unloading of modules into the Linux kernel for flexible module management.
- Learn to use the "insmod" command to insert kernel modules into the running Linux kernel, dynamically adding functionality without rebooting the system.
- Understand the "rmmod" command's role in removing kernel modules from the running kernel, allowing for dynamic unloading of modules to free up system resources and facilitate maintenance or updates.