```c
/* ANRC RHKI */
/* Lab20: Networking Lab - No ICMP */
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/in.h>

#define DRIVER_AUTHOR "ANRC"
#define DRIVER_DESC   "Lab20: Prevent Network ICMP Packets from going in/out"

MODULE_LICENSE("GPL");           // Get rid of taint message by declaring code as GPL.

/*  Or with defines, like this: */
MODULE_AUTHOR(DRIVER_AUTHOR);    // Who wrote this module?
MODULE_DESCRIPTION(DRIVER_DESC); // What does this module do?

/* IP Hooks */
/* After promisc drops, checksum checks. */
#define NF_IP_PRE_ROUTING       0
/* If the packet is destined for this box. */
#define NF_IP_LOCAL_IN          1
/* If the packet is destined for another interface. */
#define NF_IP_FORWARD           2
/* Packets coming from a local process. */
#define NF_IP_LOCAL_OUT         3
/* Packets about to hit the wire. */
#define NF_IP_POST_ROUTING      4
#define NF_IP_NUMHOOKS          5

static struct nf_hook_ops netfilter_ops_in; /* NF_IP_PRE_ROUTING */
static struct nf_hook_ops netfilter_ops_out; /* NF_IP_POST_ROUTING */

/* Function prototype in <linux/netfilter> */

/*
typedef unsigned int nf_hookfn(unsigned int hooknum,
                               struct sk_buff *skb,
                               const struct net_device *in,
                               const struct net_device *out,
                               int (*okfn)(struct sk_buff *));
*/

unsigned int main_hook(unsigned int hooknum,
                  struct sk_buff *skb,
                  const struct net_device *in,
                  const struct net_device *out,
                  int (*okfn)(struct sk_buff*))
{

        /* Check for ICMP Packet */

        /* Must be non-ICMP, let it through! */
        return NF_ACCEPT;
}

int init(void)
{
        printk(KERN_INFO "init_module() called\n");

    netfilter_ops_in.hook                  =       main_hook;
    netfilter_ops_in.pf                    =       PF_INET;
    netfilter_ops_in.hooknum               =       NF_IP_PRE_ROUTING;
    netfilter_ops_in.priority              =       NF_IP_PRI_FIRST;
    netfilter_ops_out.hook                 =       main_hook;
    netfilter_ops_out.pf                    =       PF_INET;
    netfilter_ops_out.hooknum               =       NF_IP_POST_ROUTING;
    netfilter_ops_out.priority              =       NF_IP_PRI_FIRST;
    nf_register_hook(&netfilter_ops_in); /* register NF_IP_PRE_ROUTING hook */
    nf_register_hook(&netfilter_ops_out); /* register NF_IP_POST_ROUTING hook */

    return 0;
}

void cleanup(void)
{
        nf_unregister_hook(&netfilter_ops_in); /*unregister NF_IP_PRE_ROUTING hook*/
        nf_unregister_hook(&netfilter_ops_out); /*unregister NF_IP_POST_ROUTING hook*/

        printk(KERN_ALERT "Unloading netdead ...\n");
}

module_init(init);
module_exit(cleanup);
```