تکلیف دوم عملی سیستم‌عامل

نگار سخایی

۹۵۲۸۰۰۳

برای طراحی یک ماژول کرنل برای فیلتر کردن packet های ورودی، مراحل زیر انجام شد:

۱. استفاده از کتابخانه netfilter لینوکس برای پردازش packet ها و عبور و یا drop آنان.

۲. استفاده از یک character driver در برنامه، برای ارتباط با یک کد دیگر در user space که IP:Port های موردنظر را از فایل netLKM.config می‌خواند.

```c
static int __init LKM_init(void){
    printk(KERN_INFO "Start of init!\n");
    majorNum = register_chrdev(0, DEVICE_NAME, &fops);
    if (majorNum < 0){
        printk(KERN_INFO "registering of major number failed\n");
        return majorNum;
    }
    printk(KERN_INFO "registered with major number %d!\n", majorNum);
    netLKMClass = class_create(THIS_MODULE, CLASS_NAME);
    if (IS_ERR(netLKMClass)){
       unregister_chrdev(majorNum, DEVICE_NAME);
       printk(KERN_ALERT "Failed to register device class\n");
       return PTR_ERR(netLKMClass);
    }
    printk(KERN_INFO "device class registered correctly\n");
    netLKMBevice = device_create(netLKMClass, NULL, MKDEV(majorNum, 0),
                                 NULL, DEVICE_NAME);
    if (IS_ERR(netLKMBevice)){
       class_destroy(netLKMClass);
       unregister_chrdev(majorNum, DEVICE_NAME);
       printk(KERN_ALERT "Failed to create the device\n");
       return PTR_ERR(netLKMBevice);
    }
    printk(KERN_INFO "device class created correctly\n");
    hops.hook = (nf_hookfn *) filter_func;
    hops.hooknum = NF_INET_LOCAL_IN;
    hops.pf = NFPROTO_IPV4;
    hops.priority = NF_IP_PRI_FIRST;

    nf_register_net_hook(&init_net, &hops);

    printk(KERN_INFO "initialized successfully\n");
    return 0;
```

در قسمت اول، ابتدا یک hook تعریف می کنیم که تمام پکت های ورودی، که مقصدشان سیستم ما است (NF_INET_LOCAL_IN) را دریافت می‌کند. بعد از دریافت بسته، آن را به تابع filter_func ارسال می‌کند که در آنجا، این packet، براساس IP:Port مبدا بررسی می‌شود.

```
iph = (struct iphdr *)skb_network_header(skb);
    if(!skb) return NF_ACCEPT;

    if (iph->protocol==17) {
            udph = (struct udphdr *)skb_transport_header(skb);
          //printk(KERN_INFO "Received UDP packet, ip: %pI4", &iph->saddr);
            sprintf(tbch, "%pI4:%d", &iph->saddr, ntohs(udph->source));
            int k = 0;
            for (k = 0; k < i; ++k){
                if(strcmp(tbch, ips[k])){
                    if(!mode){
                      printk(KERN_INFO
                        "Received UDP packet, ip: %s, accepted!", tbch);
                        return NF_ACCEPT;
                    }
                    else{
                      printk(KERN_INFO
                        "Received UDP packet, ip: %s, dropped!", tbch);
                        return NF_DROP;
                    }
                }
            }
            if(mode) {
              printk(KERN_INFO "Received UDP packet, %s, accepted!", tbch);
                return NF_ACCEPT;
            }
            if(!mode) {
                printk(KERN_INFO "Received UDP packet %s, dropped!", tbch);
                return NF_DROP;
            }
    }

    if (iph->protocol==6) {
```

++ قسمت مشخص شده برای زمانی ست که پروتکل لایه انتقال، udp باشد (۱۷) ولی کد مربوطه برای tcp (۶) نیز مشابه همین قسمت است. ++

IP:Port بسته با تک تک IP:Port های در لیست IP ها مقایسه شده، طبق mode تصمیم گیری ماژول، با آن برخورد می‌شود.

قسمت دیگر این ماژول که مربوط به کاراکتر device است، ورودی را از یک برنامه در user space می‌خواند و هر جفت IP:Port دریافتی را، ذخیره می‌کند.

```
static ssize_t dev_write(struct file *filep, const char *buffer, size_t len,
                                        loff_t *offset){
    if(j == 0) {
        if(strcmp(buffer, "whitelist")) mode = 0;
        else mode = 1;
        printk(KERN_INFO "Received mode %s from the user\n", buffer);
        j++;
        return len;
    }
    strncpy(ips[i], buffer, len);
    printk(KERN_INFO "Received %s from the user\n", ips[i++]);
    return len;
}
```

خط اول این فایل، mode ماژول را مشخص می کند (whitelist یا blacklist) و خطوط بعدی، هر کدام در آرایه‌ای ذخیره می‌شوند. در نهایت، زمان بررسی هر بسته، مشخصات ورودی آن بسته با همین آرایه چک خواهند شد.

```
Negar kernel: Received UDP packet 192.168.154.2:53, dropped!
Negar kernel: Received UDP packet 192.168.154.2:53, dropped!
Negar kernel: Received UDP packet 192.168.154.2:53, dropped!
```

```
Negar kernel: Received TCP packet, ip: 176.101.52.142:443, accepted!
Negar kernel: Received TCP packet, ip: 176.101.52.142:443, accepted!
Negar kernel: Received TCP packet, ip: 176.101.52.142:443, accepted!
Negar kernel: Received TCP packet, ip: 176.101.52.142:443, accepted!
Negar kernel: Received TCP packet, ip: 176.101.52.142:443, accepted!
```