Lab 11
Three Tasks

• Implement a loadable kernel module (to perform packet filtering)
• Implement:
  • stateless firewall
  • stateful firewall
Task 1 Setup

We will run the LKM in the Firewall VM and then test connectivity by running ping, telnet, etc. from the User VM.
Task 1

• Run provided code:
  • block outgoing telnet traffic

• Implement two new hook functions to block:
  • the firewall VM from pinging 8.8.8.8
  • other machines from pinging the firewall VM
unsigned int yourFilter(void *priv, struct sk_buff *skb, const struct nf_hook_state *state)
{
    // Some Parsing
    iph = ip_hdr(skb);
    
    if (some_condition) {
        return NF_DROP;
    }

    return NF_ACCEPT;
}
static struct nf_hook_ops yourFilterHook;

int addFilters(void) {
    yourFilterHook.hook = yourFilter;
    yourFilterHook.hooknum = "HOOK_NUM";
    yourFilterHook.pf = PF_INET;
    yourFilterHook.priority = NF_IP_PRI_FIRST;

    // Register the hook
    nf_register_net_hook(&init_net, &yourFilterHook);
    return 0;
}

What is \texttt{init\_net}?
A netns is a copy of the network stack:
  • With its own routes, firewall rules, etc.

Defining routing tables and firewall rules *per* netns
Network Namespace (netns)

• init_net is the initial netns in Linux

```c
149  /* Init's network namespace */
150  extern struct net init_net;
```

• nf_register_net_net_hook(&init_net, &yourFilterHook):
  • We register the hook to the initial netns
  • The rules will be applied to all traffic in this netns
Task 1: Registering the hook

```c
static struct nf_hook_ops yourFilterHook;

int addFilters(void) {
    yourFilterHook.hook = yourFilter;
    yourFilterHook.hooknum = <HOOK_NUM>;
    yourFilterHook.pf = PF_INET;
    yourFilterHook.priority = NF_IP_PRI_FIRST;

    // Register the hook
    nf_register_net_hook(&init_net, &yourFilterHook);
    return 0;
}

void rmFilters(void) {
    nf_unregister_net_hook(&init_net, &yourFilterHook);
}
```
Task 1: Initialize the module

module_init(addFilters);
module_exit(rmFilters);
Task 2 and Task 3

• Writing stateless and stateful firewall rules:
  • Protect an internal network
  • Protect internal services
  • Ensure that initiated connections from internal network are not blocked

• We will use IPMininet

• iptables: https://linux.die.net/man/8/iptables
Task 2 and Task 3: IPMininet Setup

External Network

h2

r2

r1

iptables

Internal Network

h11
h12
h13
h14
Switch-based Approach

Not used for this lab

External Network

Internal Network

iptables
## iptables: Commands

<table>
<thead>
<tr>
<th>Goal</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a table</td>
<td><code>-t &lt;table&gt;</code></td>
</tr>
<tr>
<td>Append a rule to a chain</td>
<td><code>-A &lt;chain&gt; &lt;rule_spec&gt;</code></td>
</tr>
<tr>
<td>Delete a rule from a chain</td>
<td><code>-D &lt;chain&gt; &lt;rule_spec&gt;</code></td>
</tr>
<tr>
<td>Show packet count</td>
<td><code>-L &lt;chain&gt; -v</code></td>
</tr>
<tr>
<td>List rules</td>
<td><code>-L [chain] [-t table]</code></td>
</tr>
<tr>
<td>Flush rules</td>
<td><code>-F [chain] [-t table]</code></td>
</tr>
<tr>
<td>Insert a rule to a chain</td>
<td><code>-I &lt;chain&gt; [rule-number]</code></td>
</tr>
</tbody>
</table>
## iptables: Parameters

<table>
<thead>
<tr>
<th>Goal</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>-p &lt;protocol&gt;</td>
</tr>
<tr>
<td>Jump to target</td>
<td>-j &lt;target&gt;</td>
</tr>
<tr>
<td>Input interface</td>
<td>-i &lt;intf&gt;</td>
</tr>
<tr>
<td>Output interface</td>
<td>-o &lt;intf&gt;</td>
</tr>
<tr>
<td>Source</td>
<td>-s &lt;src&gt;</td>
</tr>
<tr>
<td>Destination</td>
<td>-d &lt;dst&gt;</td>
</tr>
<tr>
<td>Extended match module</td>
<td>-m &lt;module&gt;</td>
</tr>
</tbody>
</table>
iptables: Connection Tracking

• Make sure that `conntrack-tools` is installed at your Vagrant VM
  • (Instructions in the document)

• `--cstate: NEW, RELATED, ESTABLISHED`
Installing iptables Rules in IPMininet

```
r1_rules = [Rule('-A OUTPUT -j DROP')]
r1.addDaemon(IPTables, rules=r1_rules)
```
Questions?