

# Lab 12

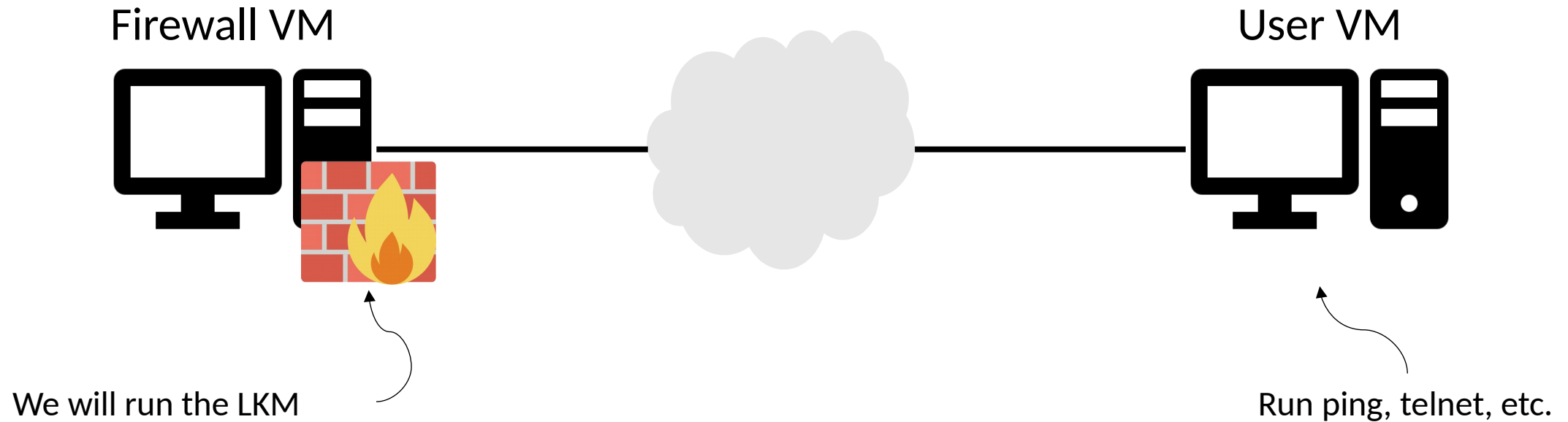
# Three Tasks

---

- Implement a loadable kernel module (to perform packet filtering)
- Implement:
  - stateless firewall
  - stateful firewall

# Task 1 Setup

---



# 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

# Task 1: Writing the Logic

---

```
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;  
}
```

# Task 1: Registering the hook

```
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;  
}
```

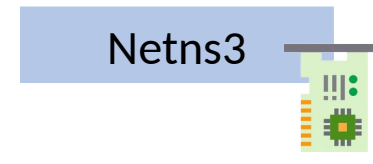
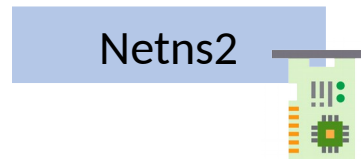
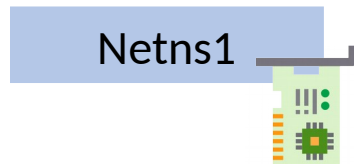
NF\_INET\_PRE\_ROUTING  
NF\_INET\_LOCAL\_IN  
NF\_INET\_FORWARD  
NF\_INET\_LOCAL\_OUT  
NF\_INET\_POST\_ROUTING

What is init\_net?

# Network Namespace (netns)

---

- 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

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

- `nf_register_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

```
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;  
}
```

NF\_INET\_PRE\_ROUTING  
NF\_INET\_LOCAL\_IN  
NF\_INET\_FORWARD  
NF\_INET\_LOCAL\_OUT  
NF\_INET\_POST\_ROUTING

```
void rmFilters(void) {  
    nf_unregister_net_hook(&init_net, &yourFilterHook);  
}
```

# Task 1: Initialize the module

---

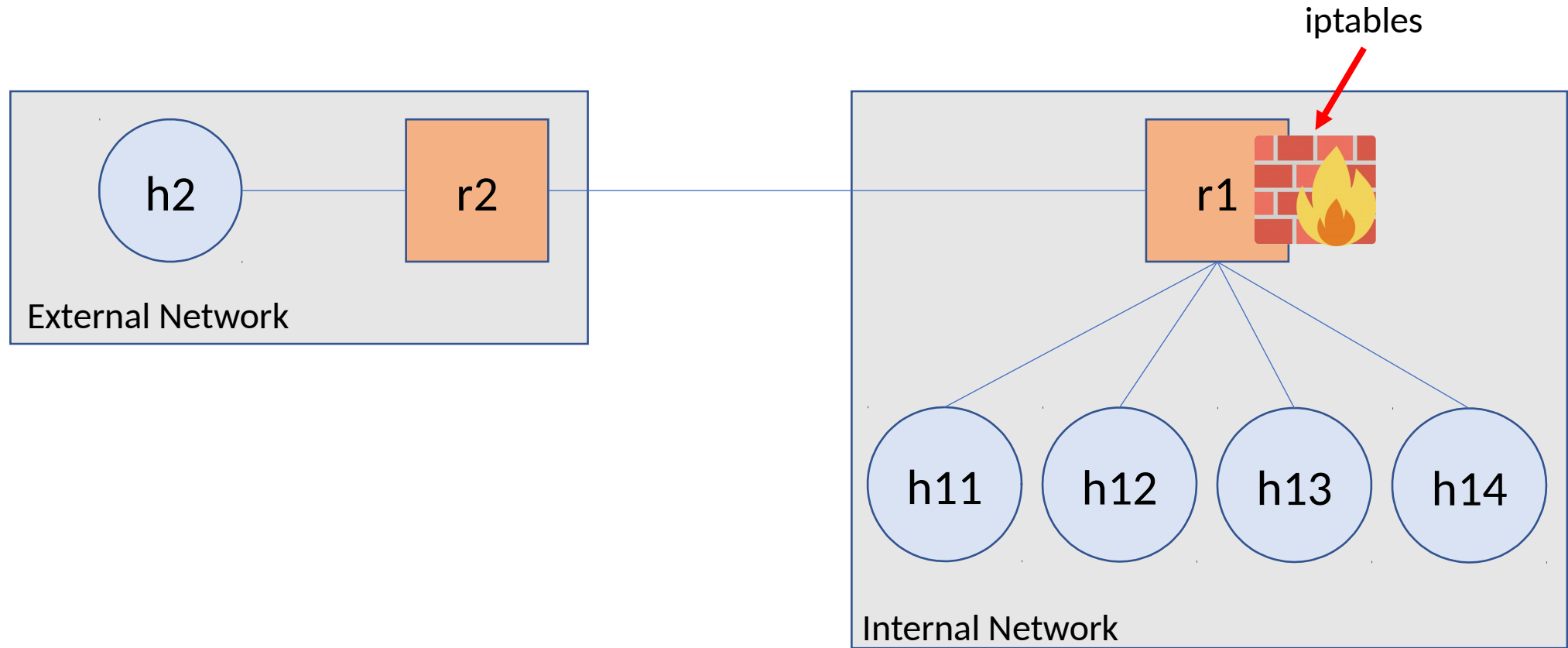
```
module_init(addFilters);  
module_exit(rmFilters);
```

# Task 2 and Task 3

---

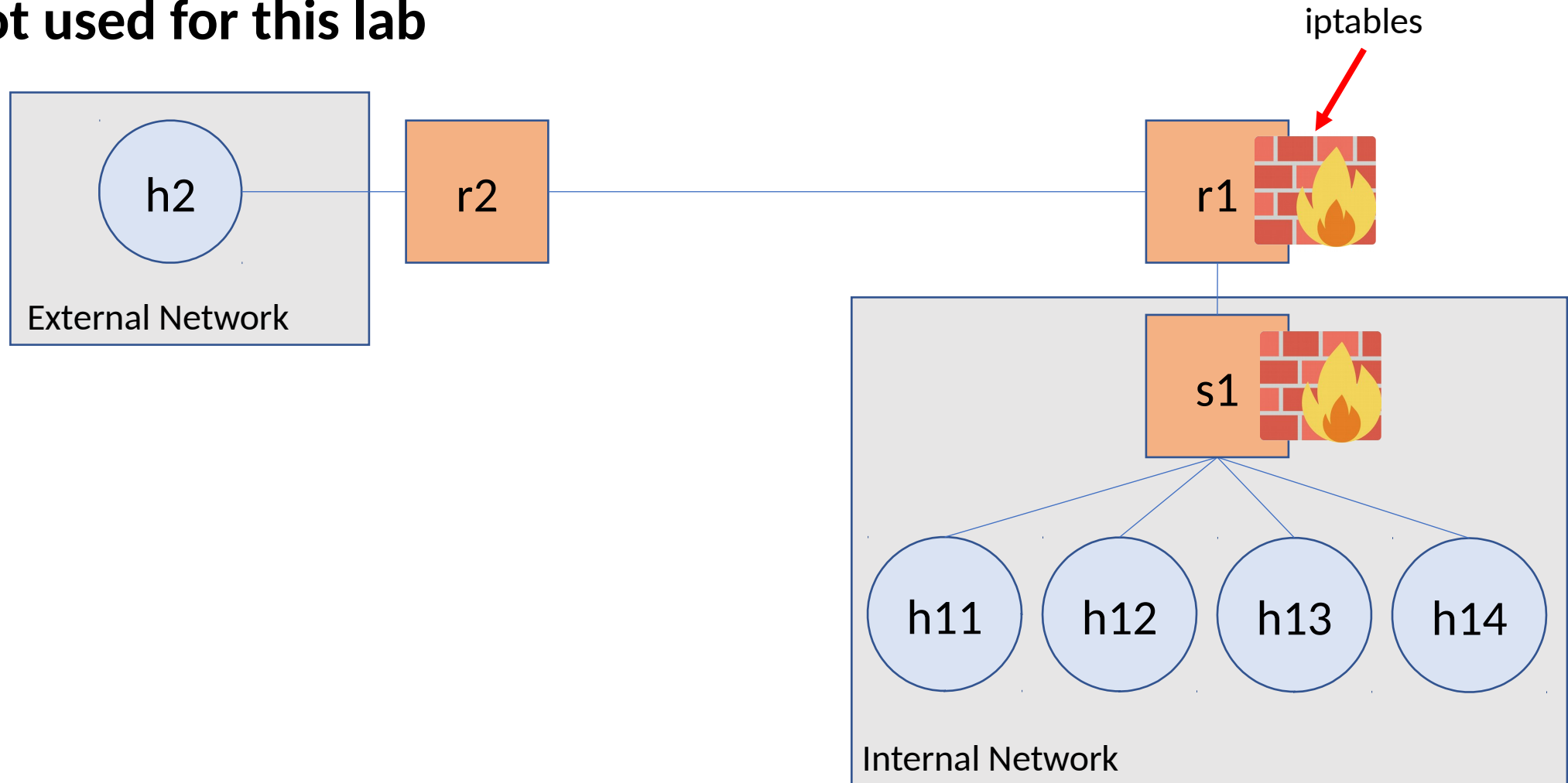
- Writing stateless and stateful firewall rules:
  - Protect an internal network
  - Protect internal services
  - Ensure that initiated connections from internal network are not blocked
- We will use IPMininet
- iptables: <https://linux.die.net/man/8/iptables>

# Task 2 and Task 3: IPMininet Setup



# Switch-based Approach

Not used for this lab



# iptables: Commands

---

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

# iptables: Parameters

---

Goal	Parameter
Protocol	-p <protocol>
Jump to target	-j <target>
Input interface	-i <intf>
Output interface	-o <intf>
Source	-s <src>
Destination	-d <dst>
Extended match module	-m <module>

# **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?

---