LAB02 How to respond against scanning

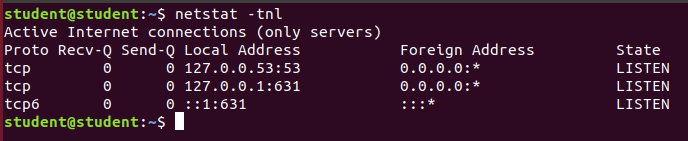
Class: M03 Student ID: B2111933 Name: Truong Dang Truc Lam

Simulation Environment

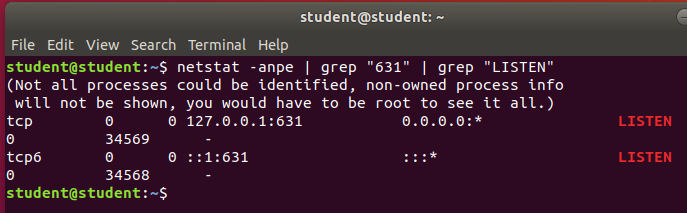
|  | Attacker | Target |
| --- | --- | --- |
| OS | Ubuntu, Window |  |
| IP | Test-bed IP |  |
| Responding type | Check the port in use  Block specific port Check the process in running | |
| Responding program | Linux command, Window command | |
| Process | [Module 1] [Module 2] [Module 3] | |

**[Module 1] Exercise 10 processes command and explain each step [essential]**

## Check the TCP port currently in use



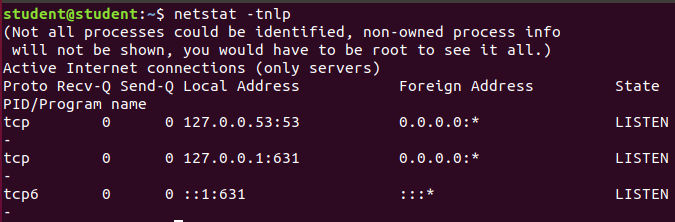
Check the TCP port currently in use



check which application is listening on a port

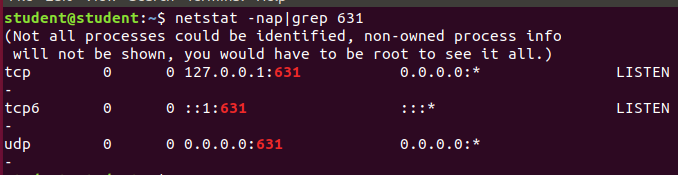
## 

## Check process currently using the port



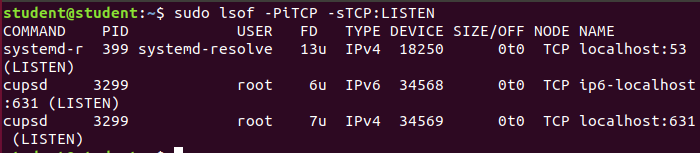
Check process currently using the port

## Confirm port number



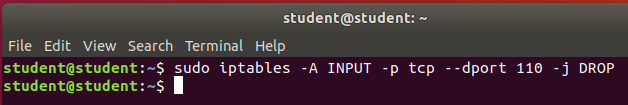
Confirm port number

## Check the port in Mac



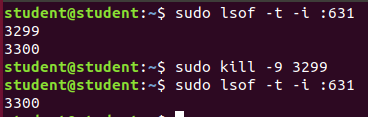
Check the port in Mac

## Block specific port



Drop pop3 port (110)

## Kill 8080 port



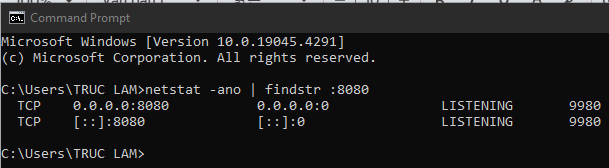
Kill 631 port (sorry my Ubuntu doesn’t use port 8080)

## Free up the Port



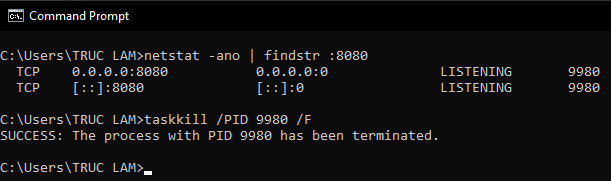
Free port 8000

## Find what service or process is running on port 8080 on Windows



I have run a Node.js server on port 8080

## Kill the process on Windows

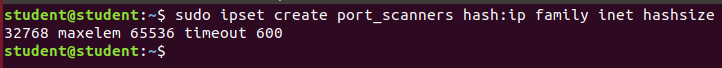


Kill the process run on port 8080 (PID = 9980)

[Module 2] Exercise 8 Linux commands and test as far as you can. Describe each test result including error status [essential]

### Create ipset liststest

* 1. $ ipset create port\_scanners hash:ip family inet hashsize 32768 maxelem 65536 timeout 600



Drop invalid incoming network packets.

* 1. $ ipset create scanned\_ports hash: ip,port family inet hashsize 32768 maxelem 65536 timeout 60



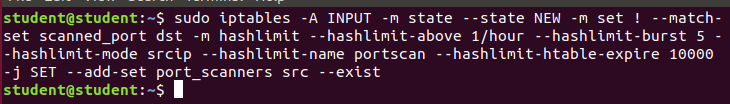
Create an IP set tracking IP-port pairs with timeout.

1. Create iptables rules
   1. $ iptables –A INPUT –m state --state INVALID –j DROP



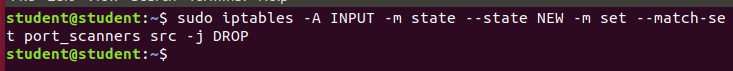
Drop invalid packets in the input chain.

* 1. $ iptables –A INPUT –m state --state NEW –m set ! --match-set scanned\_ports,dst –m hashlimit --hashlimit-above 1/hour --hashlimit-burst 5 --hashlimit-mode srcip --hashlimit-name portscan --hashlimit-htable-expire 10000 -j SET --add-set port\_scanners src –exist



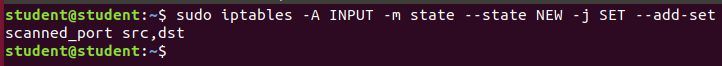
Detect potential port scans and block repeat offenders dynamically.

* 1. $ iptables -A INPUT -m state --state NEW -m set --match-set port\_scanners src -j DROP



Drop new packets from IPs in port\_scanners set.

* 1. $ iptables -A INPUT -m state --state NEW -j SET --add-set scanned\_ports src,dst



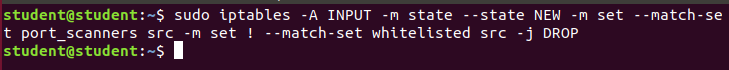
Add source-destination pair of new packets to scanned\_ports.

1. Create a whitelisted list



Create a whitelisted list

1. Change drop rule



Change drop rule

[Module 3] Port knocking commands exercise [optional]

1. Explain what is Port knocking, Port proof ?

**Port knocking** is a security technique where hidden ports open after a correct sequence of connection attempts.  
**Port proofing** ensures attackers can't exploit open ports by verifying client credentials or behavior before granting access.

1. Exercise following commands & explain each step [optional]

Step 1: Install knockd

$ sudo apt install knockd

$ sudo apt-get install iptables-persistent

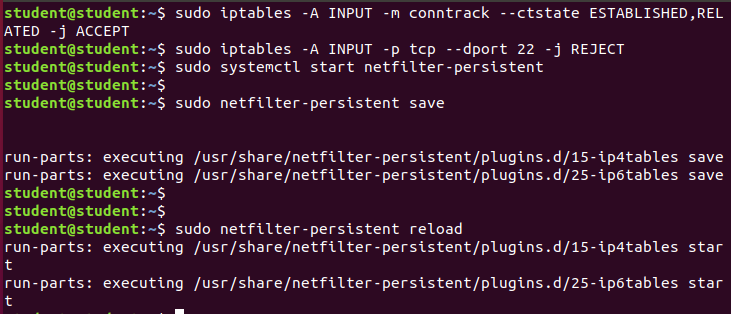
$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

$ sudo iptables -A INPUT -p tcp --dport 22 -j REJECT

$ sudo systemctl start netfilter-persistent

$ sudo netfilter-persistent save

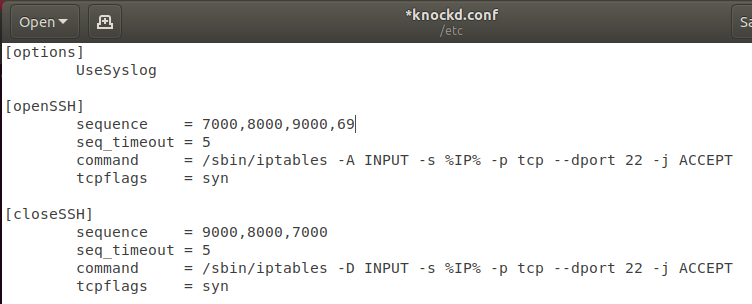
$ sudo netfilter-persistent reload



Install knockd

Step 2: Configuring knockd

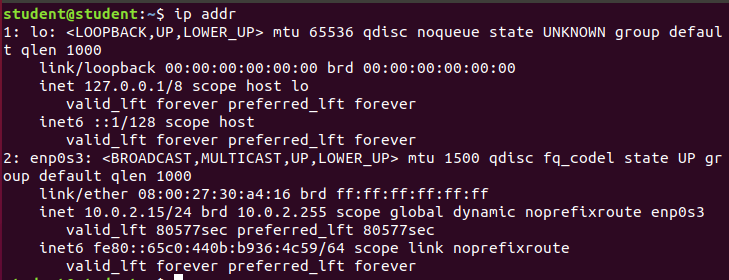
$ sudo gedit /etc/knockd.conf



Configuring knockd

Step 3: The knockd Control File Edits

$ ip addr



The knockd Control File Edits

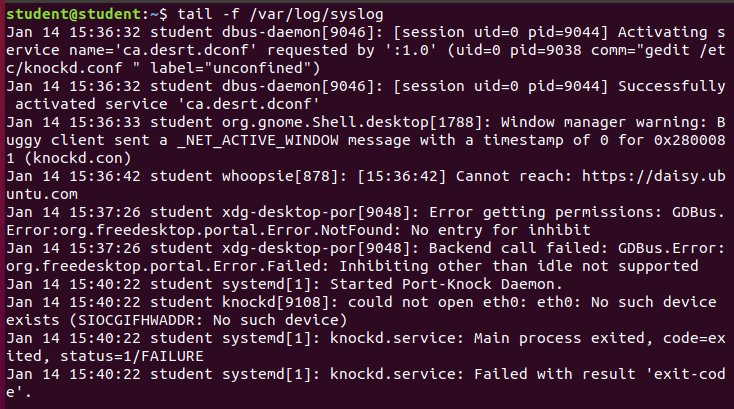
Step 4: Testing

$ sudo systemctl start knockd



Start knockd service

$ tail –f /var/log/syslog



Check log