The objective of the exercise is to build a Linux firewall and test it. This is worth 20% of your final grade, 10% for the technical work, 10% for testing and reporting.

I am issuing this as two exercises in parallel with a long deadline so that you can do it in depth, and so that it fits around my travel (for which I apologise). In past years I have given two assessed exercises successively, but changes to the project deadlines for final year undergraduates make this impractical.

You will need three Linux virtual machines, "client", "server" and "router", and two networks, "client-net" and "server-net". They should be configured as follows:

client: one network interface, 192.168.100.2/24, connected to client-net

router: two network interfaces, 192.168.100.1/24, connected to client-net and 192.168.101.1/24, connected to server-net.

server: one network interface, 192.168.101.2/24, connected to server-net.

client-net will be 192.168.100.0/24, server-net will be 192.168.101.0/24

You will need to enable ip forwarding on router.

Alternatively, you can do anything equivalent. One possibility, if you have the hardware, would be a Raspberry Pi with two VLANs into a VLAN switch. Another would be to use the Mikrotik Cloud Hosted Router (Links to an external site.) package as the router in a virtual machine. The key point is: two end points, and a router you control between them. It needs to be something I have a chance of reproducing, so please check with me if you are going to do something very different to these possibilities.

You will need to read the iptables and iptables-extensions manual pages.

Put the addresses of the machines in /etc/hosts on each machine. Install "telnet" or "nc", if they are not already installed. Install "curl" or "wget" on all machines if they are not already installed. Install a web server if it is not already installed and enable it (apache or nginx). It does not need to have any content, just return a page (even a "Not Found" page will do). Enable ssh on all machines.

Clarification: the following parts of the exercise all want you to produce a shell script or similar. In each case, the shell script should start from an empty set of rules. You can flush the rule sets with iptables -F INPUT (and similarly for FORWARD and OUTPUT), and you can reset the policy

to ACCEPT with iptables -P INPUT ACCEPT and similarly for FORWARD and OUTPUT.

Part 1: preparation (not marked)

Check you can ssh between each pair of machines ("ssh HOST date"). Check you can use curl or wget to fetch the index page between each pair of machines ("wget http://HOST/" (Links to an external site.)Links to an external site.) For reference, the six pairs are client-server, server-client, client-router, router-client, server-router, router-server).

Part 2: "default permit" host firewalling

Using the iptables command on "server", block access to port 80 so that attempts to fetch the index page time out. With server in this state, check that ssh still works from client. Write a shell script (starting "#!/bin/bash") called part2.sh which can be run on the server to achieve this result or otherwise document the process required to reproduce it.

Part 3: "default deny" host firewalling

Using the iptables command on "server", block access to all services other than port 22, so that ssh still works from client but all other ports time out. You can test this with "telnet server XX" for a random number XX from client: it should time out. Write a shell script called part3.sh which can be run on the server to achieve this result or otherwise document the process required to reproduce it.

Clarification: it should still be possible to successfully make outbound connections from the server.

Part 4: Router firewalling

Repeat parts 2 and 3, but this time using the iptables command on "router". Write a shell script called "part4-2.sh" for the repeat of part 2 and "part4-3.sh" for the repeat of part 3, which sets this up or otherwise document the process required to reproduce it.

Clarification: I have changed the names of the required shell scripts.

Part 5: BCP38 Ingress and Egress Control

Configure "router" so that "server" cannot send traffic from incorrect IP numbers, and is protected from the arrival of traffic (broadcast, RFC1918 other than the two networks we are using, source address spoofed to be "inside") which might be dangerous. Write a shell script called "part5.sh" which sets this up or otherwise document the process required to reproduce it. Note for Linux experts and those wanting to go deeper: you might like to explore one or more of blackhole routes, ipsets and the use of packet marks in the PREROUTING chain.

Part 6: Logging

Starting with the configuration you built in **part 4**, add functionality such that packets which are blocked are also logged, but with no more than one logging line being generated per second per source IP address/destination port pair. Note: do this relative to **part 4** (router firewalling) and **not** part 5 (BCP38 Control). I anticipate that testing, and therefore generating logging traffic, for part 5 will be a challenge.

Submit a zip archive containing part2.sh, part3.sh, part4.sh, part5.sh and part6.sh. Each should be able to reset then firewall to a known state and then apply the required changes. part2.sh and part3.sh run on server, part4.sh, part5.sh and part6.sh run on the router.

Part 7: testing

Write a short test plan which explains how you did the testing in parts 2–6, and include examples of your testing. Explain what you did, and how it shows that the firewall is effective. This will be separate assignment on Canvas, for which you should submit a PDF.

The assignment will be marked out of 100%: 10% for each of parts 2 through 6 (total 50%) and 50% for part 7. Testing and documenting firewalls is a crucial part of their successful use, hence the split of marks.