Fishnet Assignment 4: A Firewall

Out: Friday, February 28, 2002

Due: Tuesday, March 14, 2002.

CSE/EE461 Winter 2002; Anderson.

In this assignment, you will work in teams of two to develop a Fishnet node that implements a firewall. The program you write builds on your solution so far. The goal of this assignment is for you to understand firewalls.

1. What You Need To Write

Write a C program called hw4.c that implements a filter-based firewall with dynamic port selection, as described below. This specification may leave some points ambiguous; do what you think is best as long as your program can interoperate with the sample solution and other nodes, and document the design decisions you make.

- You will implement a firewall node that builds on top of functionality from the first three assignments:
- Takes three command line arguments as before, joins the Fishnet, performs the tasks below in any order, and runs until you type "exit", when libfish.a will end the program.
- Accepts keyboard input commands of the form "`send <nnn> <message>`" and implements the Fishnet echo protocol, which is useful for testing.
- Maintains an up-to-date routing table via the distance vector protocol and forwards packets according to the routing table and further rules specified below.
- The firewall should not implement the file transfer protocol or otherwise act as a client or server in transport connections. However, you may want to modify and reuse some small parts of your code from assignment 3.
- *Hierarchy*. In this assignment, you will be working with a hierarchical network. Nodes that share the first three digits of their addresses (e.g., 9811 and 9817) are considered to be part of the same subnet. Nodes with addresses below 1000 are not considered to be part of any particular subnet. (You can use the `fish_getsubnet` function to get the subnet of any address.) If fishhead is run with the "`--topology hierarchy`" option, then it will build a hierarchical network as follows: Nodes within a subnet are grouped together and connected pseudo-randomly, as you've seen before. The node in a subnet whose address ends in 0 (e.g., 9810) is the *border router* for that subnet. Besides being connected to other members of the subnet, it participates in an *internetwork* that connects the subnets together. Nodes with addresses below 1000 also participate in the internetwork. Note that your routing protocol should not need to change as a result of the new network topology.

- *Rules for forwarding.* The firewall is a special node that you will run in the position of border router for your subnet. The firewall's purpose is to ensure that nodes within the subnet can connect to nodes outside the subnet, but no one from the outside can connect in. The firewall should have no effect on connections that are entirely outside or inside the subnet. Therefore, the firewall needs special rules for forwarding transport packets, based on the previous hop (the frame source passed into the receive hook function) and the next hop (given in the routing table). Packets other than transport packets should be forwarded as usual.

- If both the previous hop and the next hop for packet are outside the subnet, forward the packet.

- If the previous hop is outside the subnet and the next hop is inside, drop the packet unless it belongs to an *open* connection. (See below for the definition of an open connection.)

- If the previous hop is within the subnet, forward the packet regardless of its destination. However, if its next hop is outside the subnet, the firewall must examine it to decide if it changes the set of open connections.

- *Connections*. Your firewall should be able to keep track of up to MAX_CONNECTIONS open transport connections at a time. As in assignment 3, a connection is identified by the four-tuple of source address, source port, destination address, and destination port. The source is the node inside the subnet, and the destination is the node outside the subnet.

- *Connection state*. The firewall should open a connection when it sees a SYN packet for that connection leaving the subnet. The firewall should close a connection when IDLE_TIMEOUT seconds pass without seeing a packet for the connection from the node inside the subnet. The firewall should also close the connection MAX_RETRANSMIT * RETRANSMIT_TIMEOUT seconds after it has seen acknowledgements of FIN packets sent in both directions.

- *Echo protocol*. **Optionally**, you may protect the nodes in your subnet from unwanted echo packets. The basic idea is similar to that for transport connections: When a node inside the subnet sends an echo request outside the subnet, allow the corresponding echo response through the firewall for IDLE_TIMEOUT seconds after the request was sent. Otherwise, do not allow echo requests or responses from outside the subnet into the subnet.

- *Output.* Set the debugging level in the firewall to FISH_DEBUG_APPLICATION before doing the turnin cases below. Use `fish_debug` with debug level FISH_DEBUG_APPLICATION to print the following messages when certain events occur. If the source is the node inside the subnet and the destination is the node outside the subnet, then SA is the source address, SP is the source port, DA is the destination address, and DP is the destination port.

Drop a packet	`Drop: SA:SP -> DA:DP\n`
Open a connection	`Open: SA:SP -> DA:DP\n`

Time out an idle connection	`Timeout: SA:SP -> DA:DP\n`
Close connection after seeing FINs and ACKs	`Teardown: SA:SP -> DA:DP\n`

1. Step-by-Step Development and Test Instructions

Here is a suggested set of steps to develop the required functionality.

0. Try running fishhead with the "`--topology hierarchy`" and building a network out of hw3 nodes to see how fishhead builds a hierarchical network. Or, look at the class network at http://jimbo:7777/
1. Start with hw2.c by copying it to the new file hw4.c.
2. Begin by modifying your forwarding code to drop any packets with a source outside the subnet and a destination inside the subnet. Also add the related output, since it's helpful for debugging. It may help to classify transport packets as interior (between two nodes within the subnet), exterior (between two nodes outside the subnet), outbound (from a node within the subnet to the outside), or inbound (from a node outside the subnet to within).
3. Test your classification using a fishhead configured for a hierarchical topology. Run hw4 with address 9810 (for example), to make it a border router. Then start a hw3 node with address 9811 (in the same subnet as the firewall) and another with address 1 (on the internetwork). Try to transfer a file from inside the subnet to outside the subnet and check that packets can leave the subnet, but can't enter.
4. Then, add the code to open a transport connection through the firewall and test it. You may wish to reuse or modify data structures or code from implementing transport in assignment 3 to accomplish steps 4, 5, and 6.
5. Next, add the code to close the connection at the firewall if the transport connection is idle for IDLE_TIMEOUT seconds.
6. Finally, add and test the code to close the connection at the firewall after seeing FIN packets sent and acknowledged in both directions.
7. Try introducing loss to your Fishnet. How does your firewall do?
8. Try adding a subnet with a firewall to the class network.

1. Turn In and Discussion Questions

Submit your source file(s) and the modified Makefile, if needed, using the turnin program. Hand in a paper copy of the discussion questions and test cases below as well as your source code.

1. Construct a network including a firewall, two hw3 or hw3-sample nodes belonging to the firewall's subnet, and two unfirewalled nodes. Try to transfer a file between two inside

nodes, two outside nodes, from an outside node to an inside node, and from an inside node to an outside node. Capture the output from all five nodes and turn in as your test cases. Mark up the output to indicate which output belongs to which test.

2. Leave a firewall node running on the class Fishnet, as in the previous assignments.
3. Why wait MAX_RETRANSMIT * RETRANSMIT_TIMEOUT seconds after seeing acknowledgements of FIN packets sent in both directions to close the connection, rather than closing the connection immediately?
4. In this project, we specified that the firewall should not participate as a client or server in a transport connection. In general, why might it be a good idea to disable this and other unessential services on a firewall?