This document lives here:
http://inst.eecs.berkeley.edu/~ee122/fa12/project3/project-spec-2.pdf

Project 3 - Routers, Protocols and Firewalls

Part 2

UC Berkeley, EE 122, Fall 2012

Version 1

Part 2 - Due on December 7, 2012, 11:59:59 PM

An enterprise wishes to use the plug computers as a personal firewall. You have previously, in part 1, demonstrated the efficacy of this approach. Your next task, should you choose to accept (and you better choose to accept), is to make this more realistic. In particular one would like to make it so that

1. Users connected to the plug firewall must be able to use common protocols like SSH, FTP, HTTP, HTTPS, etc. **Hence, connections to all external ports between 0 to 1023 must be allowed, and connections to all other ports must be blocked by default.**
2. We find that FTP (http://slacksite.com/other/ftp.html, http://www.ietf.org/rfc/rfc959.txt) is a very important protocol for a variety of users. However for FTP allowing connections to port 21 is not sufficient to let the FTP protocol operate correctly (*hint: because it uses multiple connections to operate!*). However, the employees must be able to use FTP to transfer files. Hence, **the firewall must appropriately unblock other ports that the FTP protocol requires for communication. Once the requirement is over, those ports must be blocked again, so that those cannot be exploited for unwanted connections.**
3. Everything not allowed by rule 1 or 2, must in fact be disallowed.

As a means to test this part of the project, we are providing you with a few FTP servers that exhibit the wide variety of behaviors one can find on the internet. Remember that real FTP servers may not behave perfectly, and you must account for this. The ones below exhibit some particularly interesting behavior.

***FTP Servers*:**

1. **ftp://ftp.mozilla.org**
2. **ftp://ftp.ibiblio.org**
3. **ftp://54.243.95.149/**

FTP Clients: All browsers have built-in FTP clients in them. The following are going to be used.

1. **Chrome** (version 14.0 or above, on any operating system)

2. **Firefox** (version 8.0 or above, on any operating system)
3. The command line utility ftp, available on all of Windows, OS X, and Linux.

All FTP client work with all FTP servers, with the exception of Firefox which will not be able to connect to **ftp://54.243.95.149/** (figure out why).

This list of FTP servers and clients has been provided for testing. Your firewall should allow any pair of FTP server and client to communicate using passive mode(s) (except those cases where a pair of FTP server and client will not work even without firewall, like the case mentioned above).

Some thoughts on this project. Please note that while we think these points might be useful as you work through your project, they might or might not be actually useful, so don't try and ascribe any particular significance to them.

- The exact messages transferred between FTP server and client may exhibit slight variances between implementations. However, all FTP servers are usually compatible with all FTP clients, so the most important fields in the message are always same. For example, command codes, response codes, and other contrl messages will always be the same, where as other text ("Entered XYZ mode", etc) in the messages may be different. Take this into consideration while parsing the messages. If you consider them correctly, then your firewall should work for any FTP server and client.
- Think carefully about the conditions and duration for allowing or blocking a port. If you allow a port for too long or allow more connections to a port than necessary for FTP, then your firewall isn't very effective. We will test your firewall to make sure it correctly accounts for such behavior, so it is extremely important you deal with them correctly.
- If you need, you can assume a connection to be over when no packets have been transferred on it for 10 seconds.

In a strengthening of the policy from the previous part, note that you will be getting only limited help from Panda (aka the TA), and other TAs. In particular, members, past and present, of the IETF (like Scott) have worked long and hard to put the internet at your disposal, and we will expect you to use that as a resource to answer some of your own questions. For instance, the FTP RFC listed above is an appropriate place to learn about FTP.

Bells-and-whistles (Extra Credit)

The bells-and-whistles require use of a new experimental version of the eecore framework. We will post the new version (keeping looking at Piazza for updates). Replace the file **/root/pox/pox/eecore/__init__.py** with the new version (do keep a copy of the old version). Note that this is an experimental version, and may have unexpected behavior. Be prepared to re-image your USB stick if you accidentally lose access to your plug.

- **BNW1**: The TAs are sometimes required to access ftp://cs.berkeley.edu/ . But browsing this FTP server can sometimes be very slow and annoying. This is because there is a misconfiguration in the firewall between the server and the internet. As an additional request, you are required to solve this issue. But you have no way to reconfigure the server or the firewall to make it faster. Instead can you do some trick using your firewall to reduce the delay? [This is a taste of real world misconfiguration problems!]

- **BNW2:** Deny downloading of files more than 1MB in size (approximately).

Tips and Tricks

Now that you have done Part 1, you are familiar with the plug as well as with the firewall's API. For doing Part 2, you can start with the following.

- If you are not familiar with using FTP through Chrome and Firefox, try it out. In the browser location bar at the top, enter "ftp://ftp.mozilla.org" as the address. You should see the list of files and folder that are present in Mozilla's FTP server.
- **Understand the FTP protocol**. FTP protocol uses 2 types of connections - control connection and data connection. The control connection is used to send commands like "list files", "transfer file", etc, and the actual data transfer takes place on the data connection. Read about the FTP protocol and figure out what ports each connection uses. A good place to start is http://slacksite.com/other/ftp.html**.** In particular, focus on the **passive mode(s)** of operation of FTP. While this site will give you the basic (and, in fact, incomplete) ideas, you are strongly advised to use Wireshark and actually see what is going on in the transferred data.
- **Understand how to use Wireshark**. This will help you take a look at TCP streams and figure out what data is transferred on the FTP connections. Install and run Wireshark on you laptop to capture packets when you do an FTP session. Make sure you capture data from the same network interface (Ethernet pr WiFi) that you are using to connect to the plug. There are quite a few resources on the Internet to learn about Wireshark. Some Youtube videos that demonstrate its use are the following.
    - http://www.youtube.com/watch?feature=player_profilepage&v=pk4OfsxxB4g
    - http://www.youtube.com/watch?feature=player_profilepage&v=nQyWWWDI_5c

  First figure out how to isolate only those TCP connections that are associated with FTP session that you have captured. Once you have isolated them, take a look at the data of each TCP stream, and the FTP requests/responses that are transferred between your laptop and the FTP server.

- By analyzing the data transferred using Wireshark, you should be able to figure out how to identify those connections that need to be allowed by the firewall for allowing FTP transfers.

Submission Details

You are required to turn in the following files -

1. *firewall.py*
2. *README.txt*
3. *firewall-extra.py*

The *README.txt* must contain the following information.

1. Your and your partner's (if any) names
2. What problems or challenges did you face in implementing your firewall?
3. Name the extra credit options you implemented (if any); describe what you have implemented, how you have implemented, and the problems you have faced while implementing it.

Cheating and Other Rules

You should not touch any other code other than firewall.py. We are aware that Python is self-modifying and therefore you could write code that rewrites the other files in the system. *You will receive zero credit for turning in a solution that modifies anything other than the firewall.py.*

The project is designed to be solved independently, but we strongly encourage work in pairs. Grading will remain the same whether you choose to work alone or with a partner; both partners will receive the same grade *regardless of the distribution of work between the two partners* (so choose a partner wisely!).

You may not share code with any classmates other than your partner. You **may** discuss the assignment requirements or your solutions -- *away from a computer and without sharing code* -- but you should not discuss the detailed nature of your solution. Assignments suspected of cheating or forgery will be handled according to the Student Code of Conduct. Apparently 23% of academic misconduct cases at a certain junior university are in Computer Science, but we expect *you all* to uphold high academic integrity and pride in doing *your own work.*