# Lab 4

For lab 4, you'll need to develop code to split messages across TCP/IP packets to evade a simple firewall that does not perform TCP/IP flow reconstruction. The following files are included in lab4files.tar.gz:

**iptables.lab4**: This the the iptables file for Ta (10.0.0.3). Ta performs a form of censorship that is similar to China's filtering of keywords in HTTP traffic. Any packets that contain blacklisted keywords are dropped and the connection is reset. The actual implementation for China's censorship does not drop the packet because they use port mirroring, and RSTs are sent in both directions, but evading both firewalls is the same. If you break the keywords in the message up across TCP/IP packets then the firewall rule does not match, but the correct message is still reconstructed on the other side of the connection. Do not try to evade these rules by sending very small packets until you have done it the right way successfully and are ready to explore other methods. At some point I'll be adding a rule to Ta to prevent small packets, and the real Great Firewall of China will refuse to tunnel small IP fragments for security reasons unrelated to the censorship. The correct way to do lab 4 is to break keywords up across packets and use large packets, you can explore other things after you're done with the basic one.

**message.txt:** This is the exact message (character-for-character) that you must pass to the server listening on your port (8080 plus your student number on 10.0.0.3) in order for the server to report success.

**serverv3.c:** This server only reports “Success!” if you pass along the exact message it is looking for. Several of the keywords in the message are censored on Ta, so you'll need to evade Ta's firewall rules in order to accomplish this. The server will timeout and exit 30 seconds after a connection is opened. If you cause your server to hang, feel free to use another port which no student has been assigned and let me know so I can kill that instance of the server and free up your port again.

I'll send an email to the class mailing list with pointers on how to forge raw TCP/IP packets that form a valid connection (including the sequence numbers, header checksums, etc.). Scapy in Python is the easiest library to use, though it has problems with dropped packets when it comes to receiving responses (for example, you'll need to grab acknowledgment numbers out of responses that come back from Ta). Libpcap in C or perl will also work. A combination of libpcap and Scapy in Python is probably the easiest way to go.

I'll give the class instructions for getting a root shell on Tha, and then you can create an account for yourself and add yourself to /etc/sudoers on Tha (10.0.0.4) and use that to do the lab by sending packets to Ta. You can set up the server code locally on your port number (8080 + your student number) on Tha and then test your code on localhost to get the basic communications, etc., working before trying it on Ta.