CPSC 526 - Assignment 3

Due date: Sunday Oct 29, 2017 @ 23:59.
Weight: 16% of your final grade.
Group work allowed, max. group size of 2.

In this assignment you are going to implement a simple port forwarding & logging program. This program could be used for many different purposes: debugging protocols, circumventing firewalls, or even as a base for man-in-the-middle applications. The program will be invoked with at least 3 arguments:

```
./proxy [logOptions] [replaceOptions] srcPort server dstPort
```

where **srcPort** is the port on which your program will listen, **server** is the name or address of a remote server, and **dstPort** is the destination port on the remote server.

Your program will listen on the given port for new connections. When a new connection is initiated, your program will open a socket to the remote server on the requested destination port. It will then relay all data from srcPort to the server's dstPort, and vice versa. Your program will also log all data to standard output. The argument **logOptions** will control the format of the logged data:

Option	**Description**
	If no logging option is given, no data logging will happen. Your program should still output information about incoming connections though.
-raw	All data will be logged as is, i.e. assumed to be ascii. Make sure to label each line of text with outgoing or incoming prefix (see examples).
-strip	Similar to the -raw option, but only printable characters will be printed. Non-printable characters will be replaced with a dot '.'. Printable characters include whitespaces.
-hex	All data will be logged in a format identical to the output of the linux 'hexdump -C' utility. This is useful for binary-only communications.
-autoN	All data segments will be divided into N-byte long chunks, and each chunk will be displayed separately, on its own line. Each byte in the chunk will be displayed based on its value. If the byte is a backslash, tab, newline or carriage return, it'll be reported in escaped form, i.e. '\\', '\t', '\n' and '\r' respectively. If the byte is in range 32..127, it will be displayed in raw form. In all other cases the byte will be displayed with a leading slash, followed by a two digit hexadecimal value of the byte.

In your output logs you must identify outgoing and incoming data by using the prefix '-->' for outgoing data, and prefix '<--' for incoming data. You must also record date/time of when the connection was initiated, as well as the source IP address of the originating connection. See the examples below for formatting details.

You also need to implement some form of replace capability, which will allow your program to search for patterns in the communication data and replace them with some other text on the fly. It should be possible to activate this replace mechanism from the command line via some [replaceOptions] switch or switches. For example:

```
$ ./proxy -raw 2001 –replace HTTP/1.1 HTTP/1.2 www.ucalgary.ca 80
```

would start the proxy just as in example 1 below, but replace all occurrences of the string “HTTP/1.1” with “HTTP/1.2” on the fly. The pattern you search for could be a simple text, or regular expression, or even a set of regular expressions. You can be creative with this part of the assignment. You will need to demo this functionality to your TA.

Additional requirements

- You may implement your program in Python, C or C++.
- You may not call external programs to help you format the output.
- You should be able to handle multiple simultaneous connections, although this is not weighted very heavily in the marking scheme.
- Once a connection has been closed, your program will wait for a new connection, i.e. your program should run forever.
- Your program should work with http, ssh, netcat-to-netcat, and your backdoor program from assignment 2.
- You only need to handle TCP sockets.

Example 1: `./proxy -raw 2001 www.ucalgary.ca 80`

This will create a tunnel from local port 2001 to University of Calgary’s web server. To test this tunnel, you can point your browser to http://localhost:2001. Sample of possible output:

```
Port logger running: srcPort=2001 host=www.ucalgary.ca dstPort=80
New connection: Sun Feb  5 15:25:57, from localhost
---> recv/send GET / HTTP/1.1
---> Host: localhost:2003
---> Connection: keep-alive
---> Upgrade-Insecure-Requests: 1
--->
<--- recv/send HTTP/1.1 404 Not Found
<--- Date: Sun, 05 Feb 2017 22:26:00 GMT
<--- Server: Apache/2.2.15 (Red Hat)
<--- Last-Modified: Thu, 15 Oct 2015 18:40:25 GMT
<--- ETag: "835f6e-349e-522290321e11b"
...
Connection closed.
```

Example 2: `./proxy -auto32 2001 localhost 22`

Similar to the example above, but now we are monitoring ssh. For example, you could try to use ssh to test this tunnel:

```
ssh localhost -p 2001
```

Here is a possible output:

```
Port logger running: srcPort=2001 host=localhost dstPort=22
New connection: Sun Feb  5 15:25:57, from localhost
---> SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ub
```

```
---> untu2.1\r\n
<--- SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ub
<--- untu2.1\r\n\00\00\054\05\14$\FF`\89\F1\CD|0\92?\DBi\03\EFk\AD\00
<--- \00\00\C4curve25519-sha256@libssh.org,
<--- ecdh-sha2-nistp256,ecdh-sha2-nis
…
```

Group work

You are allowed to work on this assignment with another student (max. group size is 2). Just beware that during the demo you will be asked to demonstrate your familiarity with all of the code. So if you do decide to group up, both of you should understand the code 100%.

Demo

You are required to demo your assignments individually. During the demo you will be asked to run your program under different scenarios, and show that your program forwards the traffic properly. You can expect to be asked to show your program to work with http, ssh, netcat-to-netcat and even with your assignment 1 backdoor. You will also need to show that your program is displaying the captured traffic. You may be asked to adjust some functionality of your program to illustrate you are familiar with the internals of the implementation. The time for your demo will be arranged by your TAs.

Submission

Submit your code via D2L. If you decide to work in a group, each group member needs to submit the assignment. You also need to submit a document (text or pdf) that includes: your name, ID and tutorial section, as well as the name of your partner (if applicable). The document must also include brief description of the functionality of your program:

- how to run it;
- which logging formats your program supports;
- sample outputs demonstrating all output formats.

Although your submission is only worth 10% of your assignment, you still need to submit this in order for your assignment to be graded at all.

Marking scheme

D2L submission, includes source code formatting & documentation	10%, required component for marking
Demo basic forwarding functionality	25%, required component for marking
Can handle multiple connections, one at a time	10%
Handles simultaneous multiple connections	10%
-raw	5%
-strip	5%
-hex	10%

-autoN	15%
Replace functionality	10%

General information about all assignments:

1. **Due time:** All assignments are due at **23:59** on the due date listed on the assignment. Late assignments or components of assignments will not be accepted for marking without approval for an extension beforehand. What you have submitted in D2L as of the due date is what will be marked.
2. **Extensions** may be granted for reasonable cases, but only by the course instructor, and only with the receipt of the appropriate documentation (e.g., a doctor's note). Typical examples of reasonable cases for an extension include: illness or a death in the family. Cases where extensions will not be granted include situations that are typical of student life, such as having multiple due dates, work commitments, etc. Forgetting to hand in your assignment on time is not a valid reason for getting an extension.
3. After you submit your work to D2L, make sure that you check the content of your submission. It's your responsibility to do this, so make sure that your submit your assignment with enough time before it is due so that you can double-check your upload, and possibly re-upload the assignment.
4. All assignments should include contact information, including full name, student ID and tutorial section, at the very top of each file submitted.
5. Although group work is allowed, you are not allowed to copy the work of others. For further information on plagiarism, cheating and other academic misconduct, check the information at this link: http://www.ucalgary.ca/pubs/calendar/current/k-5.html.
6. You can and should submit many times before the due date. D2L will simply overwrite previous submissions with newer ones. It’s better to submit incomplete work for a chance of getting partial marks, than not to submit anything.
7. Only one file can be submitted per assignment. If you need to submit multiple files, you can put them into a single container. Supported container types are TAR or gzipped TAR. No other formats will be accepted.
8. Assignments will be marked by your TAs. If you have questions about assignment marking, contact your TA first. If you still have question after you have talked to your TA then you can contact your instructor.