CPSC 526 Fall 2017 - Assignment 2

Due date: Sunday, October 15, 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 standalone backdoor program. You can write your program in C, C++ or Python. Your backdoor must be able to run on the Linux machines in the computer labs.

When your backdoor program starts, it will listen on some port for a client to connect. It should be possible to specify the port number on the command line. When the client connects, it should perform a simple hand-shake mechanism, e.g. by accepting a hard-coded password. After the handshake is successfully finished, the server will start accepting commands from the client, execute them and return results. When a client disconnects, the backdoor should resume waiting for a new connection. If the handshake is unsuccessful, i.e. the secret sent is incorrect, the backdoor should drop the connection.

Your backdoor should be usable by connecting to it using netcat (`nc`). This means that you do not need to write a client program for this assignment. Also, your backdoor program only needs to service one client at a time. There is no need to support multiple simultaneous clients.

At minimum, your backdoor must support the following commands:

Command	Description
`pwd`	return the current working directory
`cd <dir>`	change the current working directory to <dir>
`ls`	list the contents of the current working directory
`cp <file1> <file2>`	copy file1 to file2
`mv <file1> <file2>`	rename file1 to file2
`rm <file>`	delete file
`cat <file>`	return contents of the file
`snap`	take a snapshot of all the files in the current directory and save it in memory the snapshot should only include the filenames and hashes of the files the snapshot should survive client disconnecting and reconnecting later
`diff`	compare the contents of the current directory to the saved snapshot, and report differences (deleted files, new files and changed files) this does not need to be recursive
`help [cmd]`	print a list of commands, and if given an argument, print more detailed help for the command
`logout`	disconnect client (server closes the socket)
`off`	terminate the backdoor program

On top of the required commands, you must implement at least two additional commands of your own choice. You can be creative with these. Below are some suggestions:

Command	Description
`who`	list user[s] currently logged in
`net`	show current networking configuration
`ps`	show currently running processes
`nmap <params>`	run nmap with parameters <params>
`ext <program> <params>`	run program <program> with parameters <params>

Each command should give user at least a minimum feedback, eg. success or failure.

Starting the backdoor

Your backdoor should accept at least one command line argument – the port number on which the backdoor will listen. If you want to pass additional command line arguments, fell free to do so. Also, it is OK for your backdoor to print any debugging information to stdout or stderr while it's running.

```
$ ./backdoor.py 9312
backdoor listening on port 9312
```

Sample netcat session:

Once your backdoor is running, you should be able to interact with it using netcat. Here is a sample session:

```
$ nc localhost 9312
Identify yourself!
> pass p@ssw0rD
  welcome boss
> hello
  sorry, I don't understand this command
> help
  supported commands:
    pwd, cd, cwd, ls, cp, rm, mv, cat, snap, diff, off, nmap, ext
> help cp
  cp <file1> <file2> - copies file1 to file2
> cd /usr
  OK
> ls
  drwxr-xr-x.  3 root root     4096 Apr 19  2016 abrt
  -rw-r--r--.  1 root root       18 Aug 17  2015 adjtime
  -rw-r--r--.  1 root root     1518 Feb 23  2015 aliases
  -rw-r-----.  1 root smmsp   12288 Apr 19  2016 aliases.db
  drwxr-xr-x.  2 root root     4096 Aug 17  2015 alsa
> cat /etc/passwd
  root:x:0:0:Mr. Rooter Fedora Workstation,,,:/root:/usr/bin/tcsh
  bin:x:1:1:bin:/bin:/sbin/nologin
  daemon:x:2:2:daemon:/sbin:/sbin/nologin
  adm:x:3:4:adm:/var/adm:/sbin/nologin
  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
> diff
  ERROR: no snapshot
> snap
```

```
  OK
> mv passwd passwd.old
  OK
> mv aliases aliases.old
  OK
> mv passwd.old aliases
  OK
> diff
  passwd - was deleted
  aliases - was changed
  aliases.old - was added
> off
  I am terminating myself. So long...
$
```

Additional notes

- You may call external commands from your implementation. For example, you can use `system()` or `popen()` from C/C++, and `subprocess.run()` or `subprocess.Popen()` from Python.
- The "snap" command should calculate a cryptographic hash for every file in the current directory, and then store these hashes together with the filenames in memory. You can use MD5, SHA1, SHA2 or SHA3. You can compute the hashes using a library (openssl, or cryptography), or you can use an external command for this (eg. `openssl dgst -md5 -hex file`).
- The "diff" command will compare the contents of the current directory to the saved snapshot. It should be possible to call "snap" in one directory, then "cd" to a different directory, and call "diff" there. The "diff" command should only report the differences observed (deleted/added/modified files).

Group work:

Although this is a fairly simple assignment, and it was designed to be easily completed by a single person, you are allowed to work on it 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 the backdoor program on one computer, and access the backdoor from another computer. Then you will be asked to demonstrate the functionality of your backdoor. You might be asked to adjust some functionality of the backdoor program to illustrate you are familiar with the internals of the implementation. The time for your demo will be arranged by your TAs.

Submissions

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 should also include a brief description of the functionality of your backdoor:

- how to run it;
- how to connect to it;
- handshake details; and
- supported commands.

Marking

Category	**Marks**
Starting the server, being able to connect to it, disconnect from it, reconnect to it - you will receive no marks for this assignment without this basic functionality	required
Basic authentication functioning	10
Basic commands: pwd, cd, ls, cat, cp, mv, rm, help, logout, off - 5 mark penalty for any missing command - 4 mark penalty for any incorrect behavior - 3 mark penalty for missing / incorrect feedback	40
Snap / diff – report deleted files	10
Snap / diff – report new files	10
Snap / diff – report modified files	10
Snap / diff – working across different directories	5
Snap / diff – preserving snapshot through disconnect / reconnect	5
Extra command 1	5
Extra command 2	5
Style & documentation - up to 15 mark penalty for ugly code - up to 15 mark penalty for missing/inadequate documentation	
Report TXT or PDF uploaded to D2L - up to 20 mark penalty for missing information	
Code uploaded to D2L	required

General information about all assignments:

1. 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 you submit your assignment with enough time before it is due.
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: ZIP, TAR or gzipped TAR. No other formats will be accepted.

8. Assignments will be marked by your TA. If you have questions about the assignment or assignment marking, contact your TA first. If you still have question after you have talked to your TA then you can contact your instructor.