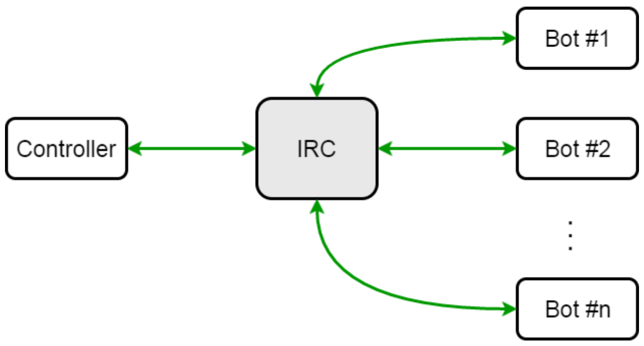
CPSC 526 – Assignment 6

Due date: Friday, December 8, 2017 at 23:59.
Weight: 16% of your final grade.
Group work allowed, maximum group size is 2 people.

In this assignment you are going to write an IRC based botnet. You are going to write both the controller program and the bot program. The purpose of the controller is to issue commands to many bots. The purpose of the bots is to accept commands from the controller and execute them. Both the controller and the bots are going to communicate with each other only through an IRC server.



The bot program

Your bot must be able to connect to an IRC server specified on the command line. It should accept a command line similar to this:

```
$ ./bot <hostname> <port> <channel> <secret-phrase>
```

The <hostname> specifies the IRC server's hostname, <port> specifies the port and <channel> specifies which IRC channel the bot will join. The <secret-phrase> specifies some secret text that the IRC bot will listen for. This secret will be used as a very basic mechanism to prevent unauthorized users to control the bot program. The bots should accept commands from a controller only after it proves the knowledge of this secret phrase.

The bot programs should be resilient. If the connection to the IRC server is terminated, or the bot is kicked, the bot should enter a loop in which it attempts to reconnect to the IRC server with a 5 second timeout. Similar behavior needs to be implemented during the initial connection – if the connection fails, the bot should attempt to connect in a loop with a 5 second timeout.

The bots will need to be able to perform a number of different functions:

- listen on the channel and detect a secret phrase from the controller
- execute commands sent by an authenticated controller to the channel
- commands will include: performing a simple network attack, migration to a different IRC server and shutting down
- report status of command execution to the controller via private messages

When you write your bot program, you should design the communication in such a way that it should be possible to control the bots by using a standard IRC client. This will make it much easier to debug your bots, even before writing the controller program. You do not have to implement any encryption for this assignment.

The controller program

The controller program will connect to an IRC server, in a manner similar to the bot. The command line arguments for the controller should be identical to the bot's command line arguments, eg:

```
$ ./conbot <hostname> <port> <channel> <secret-phrase>
```

Once the controller connects to the IRC server, it will prompt the user to enter a command. When the user enters the command, the controller will execute an action appropriate for the command. If the action involves communicating with the bots, the controller should allow some small amount of time for the bots to reply with the status. You can experiment to determine what would be a good timeout value for this, but do not make it longer than 5 seconds. During this time the controller should collect the responses from the bots, and afterwards calculate and display some useful aggregate statistic, such as total successful vs unsuccessful attacks. Once the controller is ready to accept a new command, it should prompt the user to do so.

The commands the controller needs to support are enumerated below.

Command: `status`

This command will make the controller send a message to the bots that will result in the bots identifying themselves to the controller. The controller will then print out the list of bots (their nicks), and their total number.

Command: `attack <host-name> <port>`

The controller will instruct the bots to attack the given <host-name> at the given <port> number. Every bot will connect to the given host/port and send a message containing two entries: a counter and the nick of the bot. On the next attack the counter should be increased by 1. After the attack is complete, the bots must send diagnostic messages back to the controller about the attack, i.e. success vs. failure. The controller will collect the responses from the bots, and then display them to the user on standard output. The controller will also display the total number of successful and unsuccessful attacks.

Command: `move <host-name> <port> <channel>`

This command will instruct all bots to disconnect from the current IRC server and connect to a new IRC server as specified through the arguments. The controller should display how many bots were moved as a result of this command. The controller remains connected to the same IRC server.

Command: `quit`

The controller will disconnect from the IRC and then terminate. The bots are unaffected.

Command: `shutdown`

This command will shut down the botnet. All bots should terminate. The controller will remain running, and connected to the IRC server. The controller will report the nicks of the bots that were disconnected, as well as their total number.

Example controller session

```
$ conbot irc.atw-inter.net 6667 chancpsc526 violetsareblue
Controller is running. Connected with nick: conbot663
command> status
Found 3 bots: bot1, bot2, bot3.
command> attack csx3.cpsc.ucalgary.ca 8899
bot2: attack successful
bot1: attack successful
bot3: attack successful
```

```
Total: 3 successful, 0 unsuccessful
command> attack blah.blah.blah 8899
bot2: attack failed, no such hostname
bot1: attack failed, no such hostname
bot3: attack failed, no such hostname
Total: 0 successful, 3 unsuccessful
command> shutdown
bot3: shutting down
bot2: shutting down
bot1: shutting down
Total: 3 bots shut down
command> shutdown
Total: 0 bots shut down
command> status
Found 0 bots.
command> attack csx3.cpsc.ucalgary.ca 8899
Total: 0 successfu, 0 unsuccessful
command> quit
$
```

Communication

The controller should issue its commands to the bots by posting public messages to the channel. This will make it easier for you to debug your bots by using a standard IRC client. However, the bots must communicate their responses to the controller only using private messages. This means that a random user that joins the bot channel will not see any replies from the bots, although they would see the commands issued by the controller (including the secret phrase).

Related information

A publicly accessible IRC server will be available for your testing. Information will be posted on the assignment page. However, you may decide to run your own IRC server for testing as well. Here is a link to a simple IRC server written in python:

https://github.com/jrosdahl/miniircd

To test your attacks, you can use netcat with the '-k' option. This will make netcat accept multiple connections:

```
$ nc -k -l 8963
```

You should implement the IRC communication from scratch. There are many resources available on the web related to the IRC protocol. However, you may use an existing library to help you with the IRC protocol for a 30% penalty, for example:

https://pypi.python.org/pypi/irc

IRC clients

You may want to install an IRC client to test your botnet. There are many clients available. Below are some of the more popular open source clients:

HexChat	https://hexchat.github.io	desktop client with a GUI
Kiwi IRC	https://kiwiirc.com/	web-based client
Yaaic	https://github.com/pocmo/yaaic	Android
Irssi	http://www.irssi.org/	command line client

Additional notes

- You may implement your program in Python, C or C++.
- Your program must run on the Linux machines in the labs (MS).
- You may not call external programs.
- You may be required to demo your assignments individually.
- During the demo you will be asked to demonstrate your familiarity with all of the code. If you do decide to work on the assignment in a group, both of you should understand the code 100%.

Submission

You must submit your source code and a `readme.pdf` to D2L. Please use ZIP or TAR archives. If you decide to work in a group, each group member needs to submit the assignment. The `readme.pdf` file must include:

- your name, ID and tutorial section;
- name of your group partner if applicable;
- and a section that describes how to compile and run your code.

You must submit the above to D2L to receive any marks for this assignment.

Marking

Fully functioning bot program	**60 marks**
- not handling nick collisions efficiently	-10 penalty
- not able to perform all functionality through the use of a generic IRC chat client	-10 penalty
- accepting commands from unauthenticated controller	-10 penalty
- not automatically reconnecting after disconnect	-10 penalty
- not responding to status request	-10 penalty
- not able to attack and send counter / nick in the attack message	-20 penalty
- unable to be moved to another IRC server	-10 penalty
- unable to shut down	-10 penalty
- not reporting useful feedback after every command	-5 penalty/command
- not reporting feedback via private messages	-15 penalty
- using an IRC library instead of manual implementation	-15 penalty
Fully functioning controller program (requires functioning bots!)	**40 marks**
- missing support for a command	-5 penalty/command
- not reporting feedback from bots after a command	-5 penalty/command
- missing aggregated feedback from bots	-5 penalty/command
- using an IRC library instead of manual implementation	-15 penalty
Ugly/undocumented source code	**-50 penalty**

Please format and document your source code.

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 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 is 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.