BSLE capstone User Manual

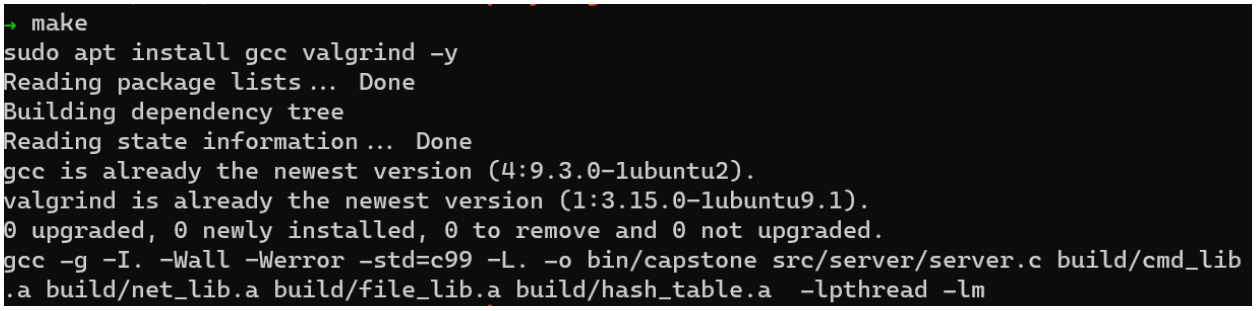
SSG Miano, Michael

Compilation:

Use either ‘make’ or ‘make valgrind’ to compile the project.

‘make’ will create required directory, and install needed libraries (gcc).

A binary called ‘capstone’ is made in the bin directory.



```
→ make
sudo apt install gcc valgrind -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version (4:9.3.0-1ubuntu2).
valgrind is already the newest version (1:3.15.0-1ubuntu9.1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
gcc -g -I. -Wall -Werror -std=c99 -L. -o bin/capstone src/server/server.c build/cmd_lib
.a build/net_lib.a build/file_lib.a build/hash_table.a  -lpthread -lm
```

Alternatively, ‘make valgrind’ compiles the server, and runs it automatically with valgrind.

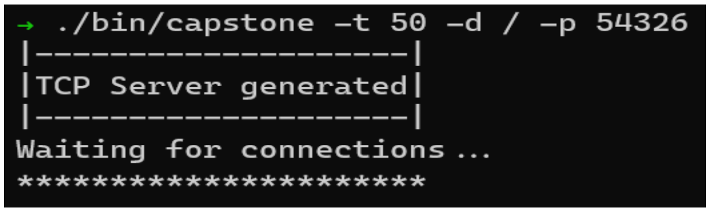
```
	valgrind --leak-check=full bin/capstone -p $(PORT) -d / -t 50
```

Note: A clean download of Ubuntu 20.04 LTS does not include gcc or valgrind by default (or make).

Starting the service:

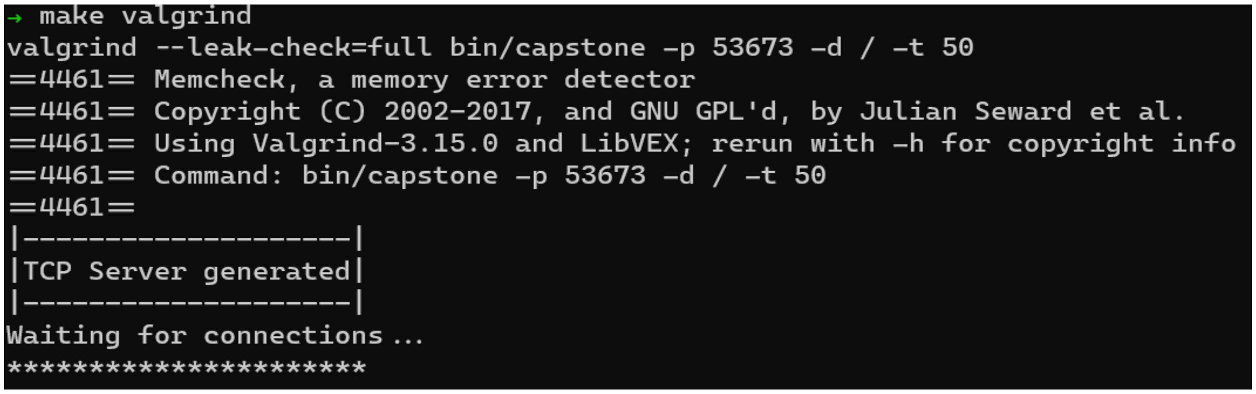
From the project root directory (where Makefile is):

- ./bin/capstone -t [timeout time] -d [server root directory] -p [port]



```
→ ./bin/capstone -t 50 -d / -p 54326
|--------------------|
|TCP Server generated|
|--------------------|
Waiting for connections...
**********************
```

Alternatively: use ‘make valgrind’ for quick start:



```
→ make valgrind
valgrind --leak-check=full bin/capstone -p 53673 -d / -t 50
==4461== Memcheck, a memory error detector
==4461== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4461== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==4461== Command: bin/capstone -p 53673 -d / -t 50
==4461==
|--------------------|
|TCP Server generated|
|--------------------|
Waiting for connections...
**********************
```

How to connect:

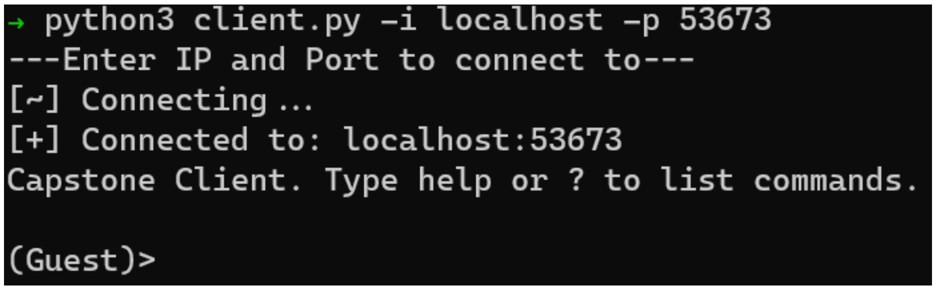
The python client is located at **src/client/client.py**

To connect to the server > cd src/client/client.py (or run it relative to your present working directory)

Note: client.py requires, at all times, two mandatory arguments: **ip address and port number**.

- ➢ python3 client.py -i [ip_address] -p [port]

If only those **two arguments** are given, the **client defaults to an interactive shell**.

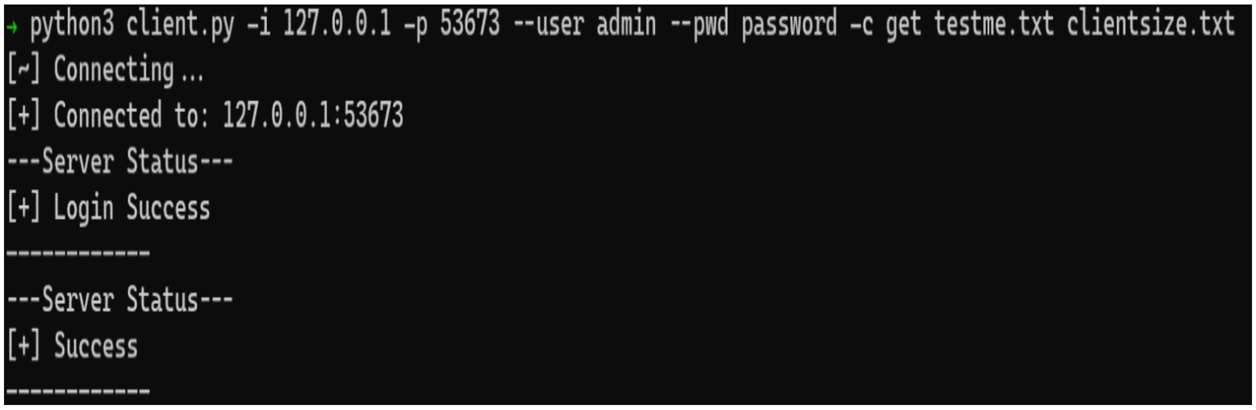


```
→ python3 client.py -i localhost -p 53673
---Enter IP and Port to connect to---
[~] Connecting ...
[+] Connected to: localhost:53673
Capstone Client. Type help or ? to list commands.

(Guest)>
```

Alternatively, when all **five arguments** are given, the **client will run in the command line interface**.

- ➢ python3 client.py -i [ip_address] -p [port] --user [username]--pwd [password] -c [cmd: opts]



```
→ python3 client.py -i 127.0.0.1 -p 53673 --user admin --pwd password -c get testme.txt clientsize.txt
[~] Connecting ...
[+] Connected to: 127.0.0.1:53673
---Server Status---
[+] Login Success
-----------
---Server Status---
[+] Success
-----------
```

Note: Both options automatically connect to the server.

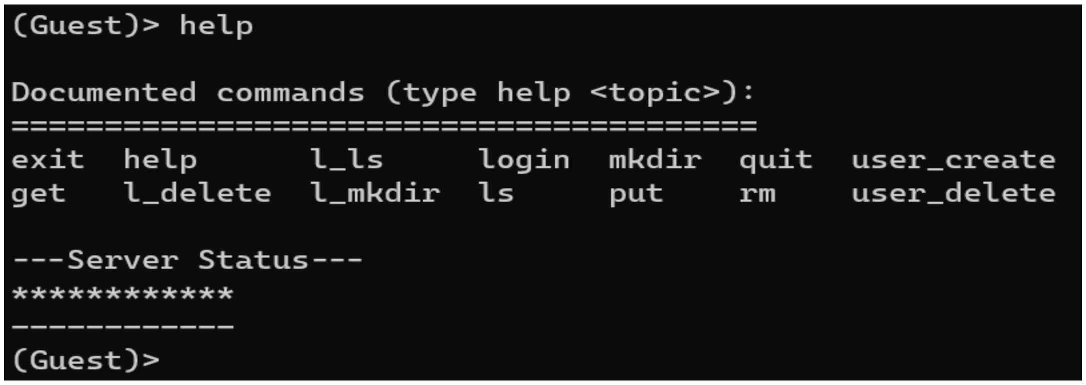
Note: The CLI option automatically attempts to login, while the interactive shell does not.

How to interact with the client:

Interactive shell:

The help menu lists all commands that the user can execute.

- ➢ help



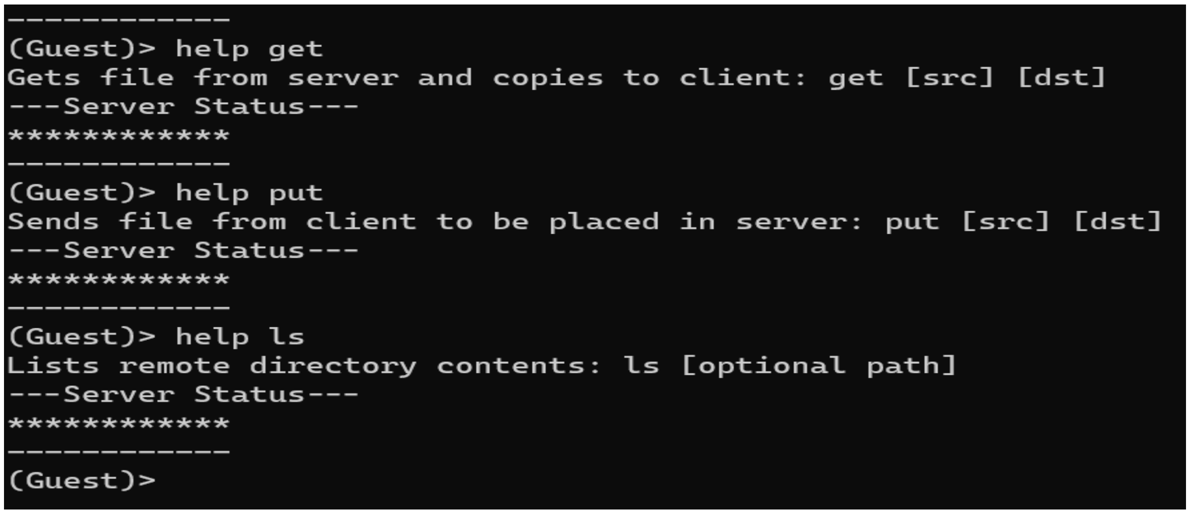
```
(Guest)> help

Documented commands (type help <topic>):
========================================
exit  help      l_ls     login  mkdir  quit  user_create
get   l_delete  l_mkdir  ls     put    rm    user_delete

---Server Status---
************
------------
(Guest)>
```

Each command can show even more help by inputting:

- ➢ help [cmd]



```
------------
(Guest)> help get
Gets file from server and copies to client: get [src] [dst]
---Server Status---
************
------------
(Guest)> help put
Sends file from client to be placed in server: put [src] [dst]
---Server Status---
************
------------
(Guest)> help ls
Lists remote directory contents: ls [optional path]
---Server Status---
************
------------
(Guest)>
```

- Commands tab-complete.
- The up arrow will populate the last inputted command.
- The shell acts like a pseudo bash shell. If a command is entered wrong, the shell will print help for that command.

Note: While the user is (Guest), no server cmds are allowed. Only local commands "l_xxx" are allowed.

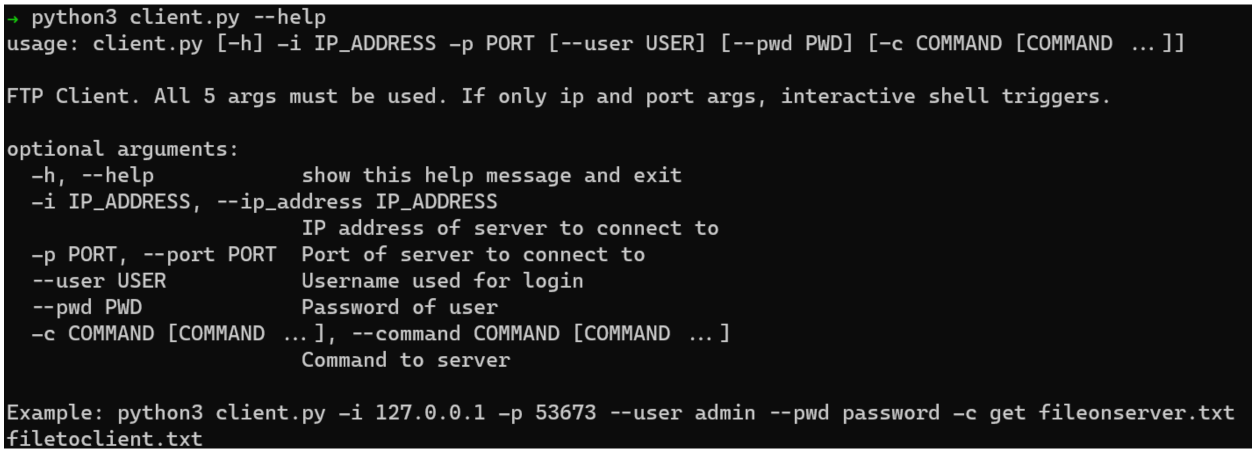
Default credentials are 'admin' | 'password'

Type 'login' and follow instructions.

CLI:

The cli help menu lists all arguments a user needs to enter.

- ➢ Python3 client.py --help



```
→ python3 client.py --help
usage: client.py [-h] -i IP_ADDRESS -p PORT [--user USER] [--pwd PWD] [-c COMMAND [COMMAND ...]]

FTP Client. All 5 args must be used. If only ip and port args, interactive shell triggers.

optional arguments:
  -h, --help            show this help message and exit
  -i IP_ADDRESS, --ip_address IP_ADDRESS
                        IP address of server to connect to
  -p PORT, --port PORT  Port of server to connect to
  --user USER           Username used for login
  --pwd PWD             Password of user
  -c COMMAND [COMMAND ...], --command COMMAND [COMMAND ...]
                        Command to server

Example: python3 client.py -i 127.0.0.1 -p 53673 --user admin --pwd password -c get fileonserver.txt
filetoclient.txt
```

Each argument shows needed values. And example is listed on at the end of the help argument.

The usage of the '-c' option is the command option, and the main workhorse.

All commands in the interactive shell are present in the CLI version (excluding quit and exit).

Note: All commands must be lowercase, (i.e. get, put, ls, etc.)

Note: For any command that requires an extra flag (i.e. Overwrite), the flag comes at the very end.

- **Example:** |… -c user_create joe_snuffy hooah RW |
  creates a user 'joe_snuffy' with password 'hooah' with read/write permissions.

Known Issues:

None currently known.