Project 1: Black-box testing

CS408, Fall 2020

Individual Project (100 points)
Due date: October 5, 2020

1 Project Description

In this project, you will do black-box testing on an FTP server. The target application is bftpd [1] in which 10 bugs have been injected. You will be provided with 10 binaries of bftpd, each with a different bug injected. In addition you will be provided with the original bftpd binary without any bugs injected. You can download all the binaries from Brightspace.

You will have to build a testing framework for bftpd that satisfies the interface described in section 2. Your testing framework must include a comprehensive test case set that tests thoroughly the bftpd binary regarding the behavior specified in section 2 and section 5 (i.e., commands not described in this document do not need to be tested).

In addition, you will need to submit a list of test cases you wrote that cause each of the buggy binaries provided to fail. These test cases must also be included with the framework you submit. Every test case in your framework must include the input and the expected output of the program.

The testing framework and the test cases for the buggy binaries that we provide are worth 80 points. The remaining points (20 points) and 10 extra points will be assigned depending on the effectiveness of your framework and test suit at finding an additional set of injected bugs that are not provided to you. Hence, you should ensure that your framework is effective at triggering and detecting a wide-range of bugs.

Read the assignment entirely before starting any work on it.

This is an individual project – collaboration between students is not allowed.

2 Testing framework interface

We provide a python skeleton code with which you can complete the project. It runs on the CS machines and uses python2. If you want to convert the skeleton code to C, or C++ you may, but if you do please provide a Makefile and detailed readme file, and have your program take the same arguments as the python program.

2.1 Testing Environment

In the skeleton code, there is a folder `testing_environment`, with two subdirectories, `testcases` and `configurations`. The testcases folder will contain, for each test case, a file name `test_input_<n>.txt` and `test_output_<n>.txt` where <n> is the number of the test case. A sample test case has been provided in the testing environment. The configuration file should contain several `.conf` files, which you can input as configuration files to your testing framework. There is an example configuration file you can use as a base, but there are 4 variables marked with TODO's you will need to change before you can use it.

The first line of each input file should be the name of a file in your configurations folder, which is the .conf file that your framework should use to test that input.

Each test input file may have to specify the input provided by different clients that connect to the server. To identify which client a line of input should be sent from, each input line in test case input should use the following format:

```
<client_num>:<input>
```

Do not forget, when we run your testing frameworks, some output may be dependent on the absolute path of the testing environment, or the user who is currently logged in, which will change when we test your project. Your framework will need to either dynamically determine what this output should be, or strategically ignore certain lines of the output. For example, the list command from the server outputs in the following format:

```
-rw-r--- 1 henry89 henry89 0 Sep 18 14:21 example2.txt
-rw-r--- 1 henry89 henry89 0 Sep 18 14:21 example.txt
```

As you can see, the file owner, group, and date modified are shown, which will change when we grade the files. Your framework can dynamically determine what the output should be, or strategically ignore values that will change when the test cases are copied to a new environment.

2.2 Running the framework

Your testing framework must implement the following interface:

```
$ python framework.py -p <binary> [-v] [-f <testcase>]
```

-v and -f are optional parameters. The -f flag is provided with the path to a test case file, and if it is provided your framework should only run that test case. The -v flag specifies whether or not your program should be providing the user with verbose output. If the verbose flag is given, your program should give detailed output for each test case, saying whether it passed or failed, as well as the expected and resulting output. If not, your program should simply output a list of which test cases failed, and which command caused each test case to fail.

You will not be able to successfully run the example skeleton until you have done the TODOs in the example configuration file.

The entire test suit must not take more than 30 minutes to execute in `data.cs.purdue.edu`. To simplify the analysis of test case results, each test case must not have more than 10 commands.

2.3 Framework Exit Status

Your framework should set its exit status to 1 if any bugs are found in the program. Otherwise, it should be set to 0. You can check that this is being done correctly by running the following command after running your framework:

```
echo $?
```

3 Setup the FTP server

- Decompress the bftpd tar file.

```
$ tar xzf bftpd.tar.gz
```

- Change PORT option in bftpd.conf if necessary. The default value is 21, but if you don't have the root privilege, you can't use any ports less than 1000. Try a random port number larger than 10000 and less than 65525 until the server works.

- Launch the buggy FTP server.

```
$ ./bftpd -c bftpd.conf -D
```

The binary will work on any 64-bit Linux machine such as data.cs.purdue.edu.

4 Connect to the FTP server

We recommend to use telnet instead of FTP client application to test the server because telnet you can send any combination of commands while FTP clients may only try valid sequences of commands.

The FTP protocol uses two types of connections: control connections and data connections. To use the FTP service, first you need to establish a control connection to the server. This connection is used to send FTP commands to the server. To upload or download a file, you need to establish a data connection. Then, when you send a command to download a file through the control connection, the server will send the contents of the file via the data connection.

In default mode, a data connection is only used for one transmission. This means a data connection is closed automatically after the transmission of one file, so if you want to download N files, you will need to open N data connections.

4.1 Control connection

When the server starts, it will listen on the port specified in the PORT option of the `bftpd.conf` file. With telnet, you can connect to the port using the following command. In `bftpd.conf`, change the FILE AUTH option to be the absolute path, a path starting from the root directory, to your `ftppasswd` file.

```
$ telnet localhost [port number]
```

If telnet fails to connect to the server, it will print out an error message. If there's no error message, telnet connected to the server and you can send commands. The following example is the sequence of commands to log in. The lines in black are FTP commands sent by the user and the lines in red are responses from the server. The commands "USER cs408" and "PASS cs408" send the username and password for login. Upon receiving a command from the user, the server sends the response, such as "331 Password please.". The first 3-digit number (e.g., 331) is the response code and the rest are messages that explain the code. More commands and expected behaviors of the server for commands are explained in subsection 4.3.

4.2 Data connection

FTP supports 2 modes, active mode and passive mode, to establish data connections. In this project, we will test only the **passive mode**. To establish connection in passive mode, a client sends a PASV command to the server through a control connection. Then the server responds with “227 Entering Passive Mode. A1,A2,A3,A4,a1,a2”. A1 to A4 are bytes of the server IP address and a1 and a2 are the port number.

```
$ telnet localhost 10010
USER cs408
331 Password please.
PASS cs408
230 User logged in.
```

The following is an example of establishing a passive data connection to the server after login.

Control	Data
(After logging in as shown above) PASV 227 Entering Passive Mode (127,0,0,1,168,179)	
	$ telnet localhost $((168 * 256 + 179)) Trying 127.0.0.1... Connected to localhost. Escape character is ’]̂’.
LIST 150 BINARY data connection established.	
	dr-xr-xr-x 13 0 0 0 Jul 26 17:57 sys drwxr-xr-x 8 0 0 4096 Aug 19 2015 mnt Connection closed by foreign host.
226 Directory list has been submitted.	

Upon receiving a PASV command, the server sends “127,0,0,1,168,179”. This means that the IP address of the server is 127.0.0.1, which is localhost, and the port of the data connection is $(168 << 8) + 179 = 43187$. In a separate session, we connect to the data port with the command “$ telnet localhost $((168*256+179))”.

After the data connection has been established, we send a LIST command that requests the list of files in the current directory. Then the server responds with the code 150 to notify that the data connection has been established and sends the contents of the directory through the data connection. The data connection is automatically closed after the transmission to mark the end. Besides the LIST command, the STOR command, which uploads a file, and the RETR command, which downloads a file, also require the data connection to be established beforehand.

4.3 FTP commands

This section explains the list of FTP commands that you should test and expected response code from the server.

For all commands The server should respond:

- 500, if the command is not recognized.
- 550, if the command is disabled with ALLOWCOMMAND XXX option in the configuration file (section 5).

USER ⟨username⟩ Send the username for login. The server should respond:

- 503, if username is already provided.
- 331, otherwise.

PASS ⟨password⟩ Send password for login. The server should respond:

- 230, if username and password are valid.
- 530, if username and password are not valid.
- 421, if new connections are not allowed. Refer to the USERLIMIT XXX option.
- 503, if no username was provided or the connection has already been logged in.

For all the following commands The server should respond:

- 503, if the connection has not been logged in.

PWD Request the current path. The server should respond:

- 257 and the path of the current working directory.

CWD ⟨pathname⟩ Request to change the current working directory to the pathname. The server should respond:

- 451 if we cannot change to the directory because of permission, invalid path and so on.
- 250 otherwise.

CDUP Request to change the current directory to the parent directory. This should be equivalent to "CWD ..".

MKD ⟨pathname⟩ Create a directory. The server should respond:

- 451 if the directory cannot be created.
- 257 and the name of the created directory and should create the directory otherwise.

RMD ⟨pathname⟩ Remove a directory. The server should respond:

- 550 if directory is not empty.
- 451 if directory cannot be removed.
- 250 and should remove the directory otherwise.

PASV Request a passive data connection. The server should respond:

- 227 and the valid IP address and port number for the data connection.

NLST Request a list of filenames in the given directory (defaulting to the current directory), with no other information. Must be preceded by a PASV command.

LIST [⟨pathname⟩] Request the list of the files in the directory. If pathname is not given, it is considered to be the current working directory. The server transfers the list through the data connection. The server should respond:

- 425 if the data connection has not been established.
- 150 if the data connection has already been established.
- 226 when the transfer has been finished.

STAT ⟨pathname⟩ Returns general status information regarding the given filename or directory (data is sent over the control connection, no PASV command required).

SIZE ⟨pathname⟩ Returns the size of the given file or directory.

STOR ⟨pathname⟩ Create a file with pathname and store the content that will be transferred through the data connection into the file. The server should respond:

- 425 if the data connection has not been established.
- 553 if the server cannot open the file.
- 150 if the data connection has already been established.
- 226 when the file transfer has been finished.

RETR ⟨pathname⟩ Retrieve a file with pathname from the server. The server sends the content of the file through the data connection. The server should respond:

- 425 if the data connection has not been established.
- 553 if the server cannot open the file.
- 150 if the data connection has already been established.
- 226 when the file transfer has been finished.

REST ⟨byte-count⟩ Set the offset for STOR and RETR command. For example, the sequence of commands "REST 10" and "RETR fileA" should retrieve the fileA from the 11th byte. After a STOR or a RETR command, the offset should reset to 0. The server should respond 350.

5 Configuration file

In order to find bugs, you may need to try different values for the configuration options for the FTP server. The configuration files consist of several sections. First, there is a global `section global{ ... }`. The options in this section applies to the entire server and users. Next, there are sections for each user such as `user root { ... }`. The values in this section only apply to the user. The comment in the configuration file explains the meaning of each option as well.

PORT option This option specifies the port number for the control connection. If you are working on a shared machine such as `data.cs.purdue.edu`, you need to change the value to a random value between 10000 and 65535.

DENY LOGIN option If the value is other than "no", the message should be sent to the client when the user is trying to login.

FILE AUTH option This file contains the usernames and passwords.

ALLOWCOMMAND XXXX option If the value is "yes", the server should respond with 550 if a client sends the command `XXXX`. For example, if ALLOWCOMMAND STOR is set to yes, the server should respond 550 whenever clients send the STOR command.

HELLO STRING and QUIT MSG options After connecting to the session, the server should respond with the message in the HELLO STRING option. When the client sends QUIT command, the server should respond with the message in QUIT MSG option.

USERLIMIT GLOBAL option If the value is other than "0", limit the number of control connections at a time. If the limit is reached, the server should respond with 421 when a new client is trying to login.

USERLIMIT SINGLEUSER option If the value is other than "0", limit the number of control connections of a single user at a time. If the limit is reached, the server should respond with 421 when a new client is trying to login.

6 Submission

You must submit your work in a **single zip** file using Brightspace with the following filename format, making sure to replace *username*, *firstname* and *lastname* with the appropriate content:

username_firstname_lastname.zip

Make sure you use the skeleton provided for your submission. Your submission must include the skeleton directory provided, with the code for your framework in the main directory, and your tests must be provided in the testing_environment folder as described in 2.1.

The submission must satisfy the following requirements:

- All code must be written in **C or python exclusively**. No other languages will be allowed.
- Your main program file should be named framework.c, or framework.py
- Your test cases must follow the layout desribed in section 2.1

- The framework must not rely on any external tool. The only allowed resources are the standard libraries included with python and C.
- Your submission must only include your own code. Do not consult or use any other code. Refer to the course policy on plagiarism for additional details about the course policy.

References

[1] RFC959: File Transfer Protocol. https://www.ietf.org/rfc/rfc959.txt

[2] BFTPd. http://bftpd.sourceforge.net