Computer Networking
Università della Svizzera italiana — S. Santini, L. Alecci, M. Laporte

Graded Assignment n. 2
April 27, 2022

Assignment 2: Computer Networking Capture The Flag!

Due date: *Wednesday, May 11, 2022 at 23:59*

This is a group assignment. You may discuss it with other groups, as long as you do not copy their code nor documentation. You must properly reference any and all sources you used. Information about grading and submission guidelines are available throughout and at the end of the assignment (Sections 2.3 and 2.4).

Changelog

v1 (27.04.2022): first published version.

1 Introduction

εκμεταλλεύομαι

Capture-The-Flag - hereinafter **CTF** - are hacking competitions in which the participants try to exploit services vulnerabilities. Each exploited service gives a *flag* (an alphanumeric string) to the team that breaks the security of the services of another team. After obtaining a flag, the attacker team submits it to a so called *scorebot service* in order to receive the points associated to the specific challenge.

In this assignment we were inspired by CTF competitions and we have implemented 10 network challenges in what we could call Computer Networking CTF. The assignment request is to solve these 10 challenges and submit their flags to a scorebot.

2 Rules

Shortcuts

Any attempt of plagiarism, hacking and/or brute-forcing the server will be **severely** punished.

2.1 Requirements

- The assignment **MUST** be solved using Python, versions 3.x.
- Only standard and custom (i.e. made by you) modules are allowed
- The solution of each challenge **MUST** be in different files, named client_x.py, where x is the challenge number.
- Each team member **MUST** submit their flags.
- Different teams **CANNOT** share flags.

2.2 Output

Each solution **MUST** print to the screen the content of every message exchanged between client and server (in string format, **do not** print bytes). More details are described in the Implementation section.

2.3 Submission

Each group can submit their solution either uploading a compressed zip/tar file to iCorsi, or by submitting a txt file with the URL to a GitHub repository. If you chose the GitHub way, you **must** add the user **petemir** to your project.

2.4 Grading

- 50% of the final grade will be assigned automatically using the flags sent to the scorebot.
- 25% of the grade will be determined using automatic test of your output.
- 25% of the grade will be assigned manually by reviewing your code.

3 Implementation

3.1 Setup

The server is located in the internal network of USI. Therefore, in order to reach it, you have to connect to the USI VPN.

If you need to connect to the USI VPN follow the steps detailed in the following link: https://usi.4me.com/self-service/search?q=VPN

Check that your VPN client is connected when you work on the assignment, otherwise you will not be able to reach neither the server nor the scorebot.

3.2 Challenges

Each solved challenge gives the team **1 point** for the final grade.

1 - First flag with TCP

- Connect to 162.243.73.199:9990 using TCP
- The server will send you your first flag
- Send the flag to the scorebot

Output example

```
/> python client_1.py
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

2 - First flag with UDP

- Create a socket able to send and receive UDP datagrams to 162.243.73.199:9991
- Send the string ”*helo*” to the server. Use lowercase!
- The server will send you the flag
- Send the flag to the scorebot

Output example

```
/> python client_2.py
helo
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

3 - Random port

- Connect to 162.243.73.199:9992 using TCP
- The server will send you <*random_port*>, an integer, which is the port number that you must use in the next connection. You have less than 2 seconds to reply, so **do not** do it manually!
- Connect to 162.243.73.199:<random_port>and get your flag
- Send the flag to the scorebot

Output example

```
/> python client_3.py
random port: 13089
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

4 - Estimated RTT

- Connect to 162.243.73.199:9993 using TCP
- The server will send you three values: the current estimated round-trip-time, an alpha value, and the last RTT
- Send back the value of the new estimated RTT (rounder to the nearest integer!) using the exponential weighted moving average calculation seen in class
- If the value is correct the server will send you the flag otherwise it will return you an error
- Send the flag to the scorebot

Output example

```
/> python client_4.py
161ms 0.088 278ms
171
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

5 - Transmission delay

Suppose there is exactly one switch between a sending host and a receiving host. The transmission rates between the sending host and the switch and between the switch and the receiving host are, respectively, R1 and R2. Assuming that the switch uses store-and-forward packet switching, what is the total end-to-end delay to send a packet of length L? Ignore queuing, propagation and processing delays.

- Connect to 162.243.73.199:9994 using TCP
- The server will send you the following values:
  - L = length of the packet

  - R1 = Transmission rate of the first link
  - R2 = Transmission rate of the second link

- Submit the total transmission delay as requested in the midterm (in seconds, avoid to send the s character as shown in the example below)
- Send the flag to the scorebot

Tip: the server will only send you these units: {bits, Kb, Mb, Gb, Kbps, Mbps, Gbps}

Example:

1. client connects to the server
2. client ⇐ server: L=128 bits R1=1 kbps R2=2 kbps
3. client ⇒ server: 0.192
4. client ⇐ server: *flag*

Output example

```
/> python client_5.py
L=1000 Gb R1=1 Gbps R2=1 Gbps
2000.0
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

6 - Simplified TCP connection establishment using strings

In this challenge we will simulate the TCP 3-way handshake, used to initialize a TCP connection between two hosts. We will use strings.

- Connect to 162.243.73.199:9995
- Send the <SYN>string with a Seq value equal to zero
- Receive the <SYN/ACK>string
- Send the <ACK>string
- If the values are correct you will receive the flag, otherwise you will receive an error
- Send the flag to the scorebot

Example:

1. client connects to the server
2. client ⇒ server: SYN Seq=0
3. client ⇐ server: SYN,ACK Seq=0 Ack=1
4. client ⇒ server: ACK Seq=1 ACK=1
5. client ⇐ server: *flag*

Output example

```
/> python client_6.py
SYN Seq=0
SYN,ACK Seq=0 Ack=1
ACK Seq=1 Ack=1
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

7 - Simplified TCP connection close using strings

In this challenge we will simulate the TCP 4-way handshake, used to close a TCP connection. We will use strings.

- Connect to 162.243.73.199:9996
- Send the <FIN,ACK>string with random data of Seq and Ack
- Receive the <ACK>string
- Receive the <FIN,ACK>string
- Send the <ACK>string
- If the values are correct you will receive the flag otherwise you will receive an error
- Send the flag to the scorebot

Example:

1. client connects to the server
2. client ⇒ server: FIN,ACK Seq=5 Ack=9
3. client ⇐ server: ACK Seq=9 Ack=6
4. client ⇐ server: FIN,ACK Seq=9 ACK=6
5. client ⇒ server: ACK Seq=6 Ack=10
6. client ⇐ server: *flag*

Output example

```
/> python client_7.py
FIN,ACK Seq=1 Ack=5
ACK Seq=5 Ack=2
FIN,ACK Seq=5 Ack=2
ACK Seq=2 ACK=6
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

8 - IPv4 Subnetting

In this challenge, the server will give us an IP address, a network mask, and we will need to **calculate** (i.e., **do not use libraries that do this for you**) the network address.

Tip: you can use https://www.calculator.net/ip-subnet-calculator.html to understand how to calculate a network address.

- Connect to 162.243.73.199:9997
- Receive an IP address and network mask
- Send the network address
- If the values are correct you will receive the flag otherwise you will receive an error
- Send the flag to the scorebot

Output example

```
/> python client_8.py
51.128.82.195 30
51.128.82.192
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

——————————————————— Απο εδώ γίνεται δύσκολο ———————————————————

9 - IPv4 Routing

In this challenge, the server will send us the routing table of a router, and a destination IP. We need to find which interface will be used when packets are sent to that IP.

- Connect to 162.243.73.199:9998
- Receive a list of network address with their corresponding interfaces, and a destination IP
- Send the interface that will be used to send packets to that IP
- If the values are correct you will receive the flag otherwise you will receive an error
- Send the flag to the scorebot

Tip: look the flag size before sending it to the scorebot.

Output example

```
/> python client_9.py
144.0.0.0/8 0,200.0.0.0/11 1,otherwise 2,25.1.130.184
2
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

10 - Internet Checksum

- Connect to 162.243.73.199:9999
- Receive the string representation of the first and second byte
- Send back the Internet checksum as string
- If the Internet checksum is correct you will receive the flag otherwise an error
- Send the flag to the scorebot

Tip: https://code4coding.com/python-example-to-sum-of-two-integer-using-bitwise-operator/.

Output example

```
/> python client_10.py
10001000 00100010
01010101
aaaaabbbbbccccca0501a3f787230abf579f6db2dd55be0fa3450f8acd54e6f3
```

11 - Bonus (2 points)

Submit the flags automatically inside each client or in a separate program. The points for this bonus challenge will be assigned only if the user has at least 8 flags correctly stored in the database.

3.3 Scorebot

The flags are 64 characters alphanumerical strings.

To send a flag:

1. Connect to 162.243.73.199:11111 using TCP
2. Send the following string: *<your_username><challenge_number><flag>*
3. The scorebot will reply you accordingly if the flag was accepted or not

- *<your_username>*: represents a string composed by the concatenation of your name and lastname (both lowercase) where each space is substituted with a dot. Each team member must submit their flags. Example: "Matias Laporte" ⇒ "matias.laporte"
- *<challenge_number>*: the challenge number, as described in this document, for which you are sending the flag.
- *<flag>*: 64 alphanumeric characters. They represent the flag.

Everytime you send a flag for a specific challenge you will overwrite its value for the specific challenge.

Example:

1. first attempt: matias.laporte 3 hgur7iks9pl477hynzTT
2. second attempt: matias.laporte 3 AAAAAAAAAAAAAAAAAAAA

The final flag associated to **challenge 3** for the user **matias.laporte** will be **AAAAAAAAAAAAAAAAAAAA**.

4 Tips

- Use the utf-8 charset if you need to encode messages.
- All of the services expect a fast reply. Do not try to solve problems manually: you need to implement a Python solution for each challenge.
- If you have to print a bytes type use the `decode()` function to convert it to a string type
- Use `print(variable, end='')` if you do not need to print an extra newline character

Bugs? Please send us an e-mail :).

5 Questions and Answers

Q: Can team members send the same flag?

A: Yes. You have two options: each member sends the same flag, or you run your client 3 times (flags are not unique!), gather 3 flags, and each member sends a different one. But remember that **each member has to submit a flag for each exercise**.

Q: Can we use the same flag for different challenges?

A: No, the scorebot will tell you that the flag is invalid :).

Q: How can we know if my flag was correctly saved?

A: The score will reply accordingly, saying if the flag was correctly saved, or if it was invalid.

Q: Can we use a shell script for submitting the flags to the scorebot?

A: Yes, but this approach would not earn you the extra points.

Q: Can I change usernames between challenges?

A: No, your username should be unique if you follow the naming rules :).

Q: Can I cheat? (using the code from another team, sharing flags among teams, using brute-forcing attempts, etc.)

A: You still have to provide the python code for your solution, so why would you try to cheat when you have to submit the code anyway :(. Furthermore, as in the first assignment, the solutions will be compared between teams.