Computer Network
Southern University of Science and Technology
Mengxuan Wu
12212006

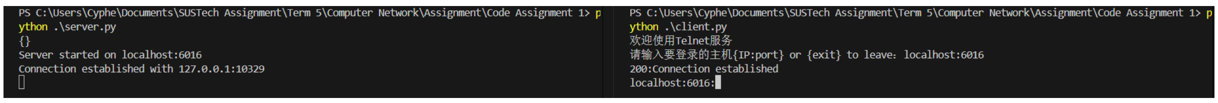
Assignment 1

Mengxuan Wu

1 Demonstration of the program

In the following sections, figures of CLI are shown to demonstrate the program. If two CLI are opened, the left one is the server and the right one is the client. If only one CLI is opened, it is the client.

1.1 Task 1

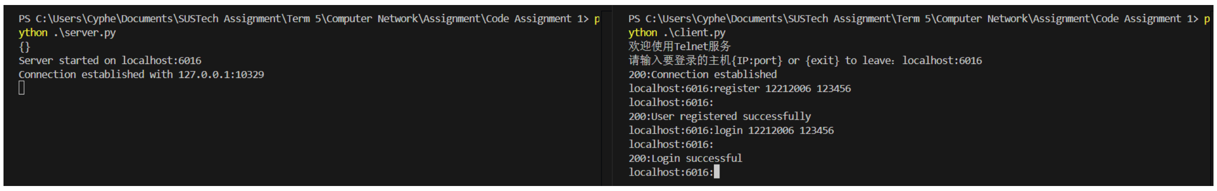


```
PS C:\Users\Cyphe\Documents\SUSTech Assignment\Term 5\Computer Network\Assignment\Code Assignment 1> p
ython .\server.py
{}
Server started on localhost:6016
Connection established with 127.0.0.1:10329
```

```
PS C:\Users\Cyphe\Documents\SUSTech Assignment\Term 5\Computer Network\Assignment\Code Assignment 1> p
ython .\client.py
欢迎使用Telnet服务
请输入要登录的主机{IP:port} or {exit} to leave: localhost:6016
200:Connection established
localhost:6016:
```

Figure 1: Initialization of the server and client

1.2 Task 2



```
PS C:\Users\Cyphe\Documents\SUSTech Assignment\Term 5\Computer Network\Assignment\Code Assignment 1> p
ython .\server.py
{}
Server started on localhost:6016
Connection established with 127.0.0.1:10329
```

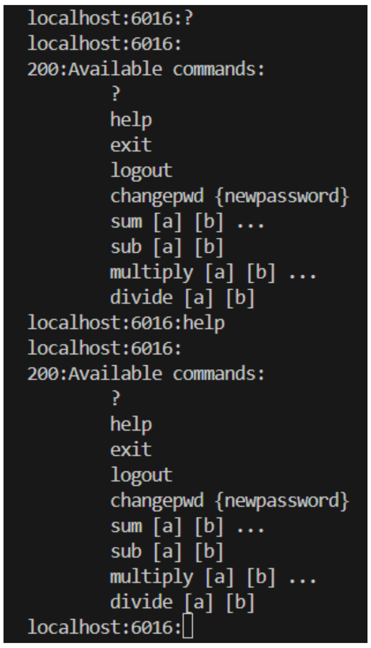
```
PS C:\Users\Cyphe\Documents\SUSTech Assignment\Term 5\Computer Network\Assignment\Code Assignment 1> p
ython .\client.py
欢迎使用Telnet服务
请输入要登录的主机{IP:port} or {exit} to leave: localhost:6016
200:Connection established
localhost:6016:register 12212006 123456
localhost:6016:
200:User registered successfully
localhost:6016:login 12212006 123456
localhost:6016:
200:Login successful
localhost:6016:
```

Figure 2: Register and login

1.3 Task 3

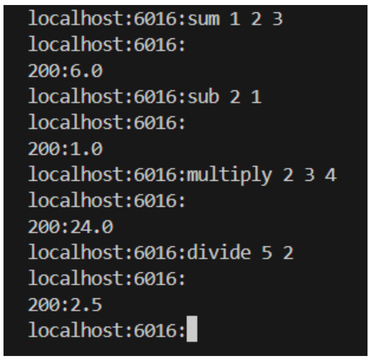
No image since all implementation is in the code.

1.4 Task 4



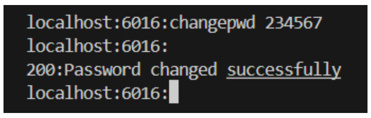
```
localhost:6016:?
localhost:6016:
200:Available commands:
        ?
        help
        exit
        logout
        changepwd {newpassword}
        sum [a] [b] ...
        sub [a] [b]
        multiply [a] [b] ...
        divide [a] [b]
localhost:6016:help
localhost:6016:
200:Available commands:
        ?
        help
        exit
        logout
        changepwd {newpassword}
        sum [a] [b] ...
        sub [a] [b]
        multiply [a] [b] ...
        divide [a] [b]
localhost:6016:
```

Figure 3: Help command



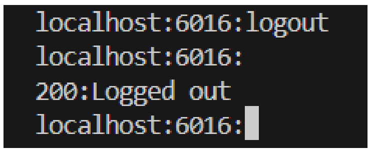
```
localhost:6016:sum 1 2 3
localhost:6016:
200:6.0
localhost:6016:sub 2 1
localhost:6016:
200:1.0
localhost:6016:multiply 2 3 4
localhost:6016:
200:24.0
localhost:6016:divide 5 2
localhost:6016:
200:2.5
localhost:6016:
```

Figure 4: Arithmetic command



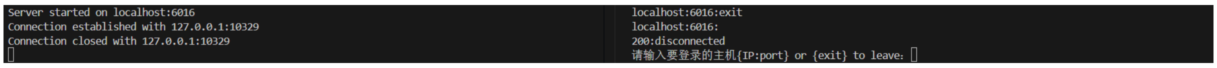
```
localhost:6016:changepwd 234567
localhost:6016:
200:Password changed successfully
localhost:6016:
```

Figure 5: Change password command



```
localhost:6016:logout
localhost:6016:
200:Logged out
localhost:6016:
```

Figure 6: Logout command

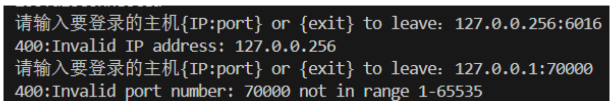


```
Server started on localhost:6016
Connection established with 127.0.0.1:10329
Connection closed with 127.0.0.1:10329
```

```
localhost:6016:exit
localhost:6016:
200:disconnected
请输入要登录的主机{IP:port} or {exit} to leave:
```

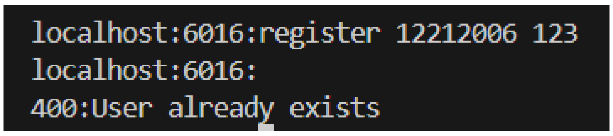
Figure 7: Exit command

1.5 Task 5



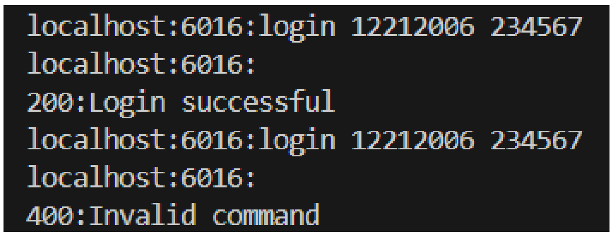
```
请输入要登录的主机{IP:port} or {exit} to leave: 127.0.0.256:6016
400:Invalid IP address: 127.0.0.256
请输入要登录的主机{IP:port} or {exit} to leave: 127.0.0.1:70000
400:Invalid port number: 70000 not in range 1-65535
```

Figure 8: Invalid IP or port



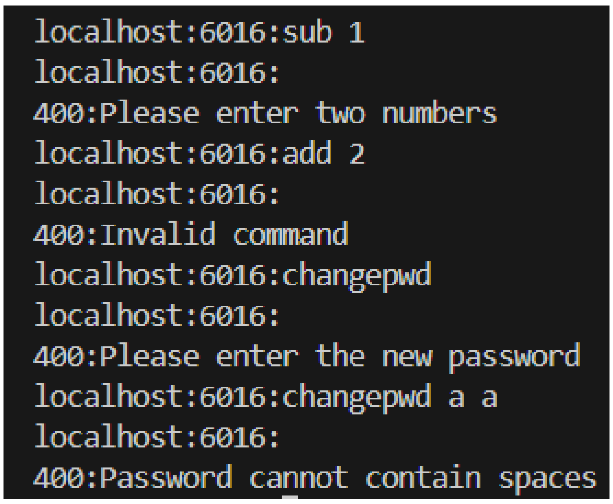
```
localhost:6016:register 12212006 123
localhost:6016:
400:User already exists
```

Figure 9: Attempt to register with an existing username



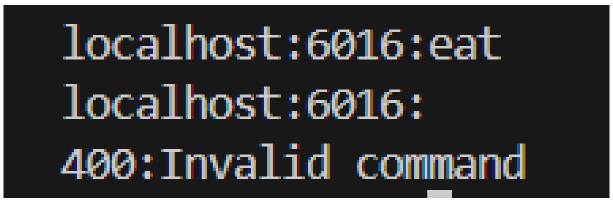
```
localhost:6016:login 12212006 234567
localhost:6016:
200:Login successful
localhost:6016:login 12212006 234567
localhost:6016:
400:Invalid command
```

Figure 10: Attempt to login again after logging in



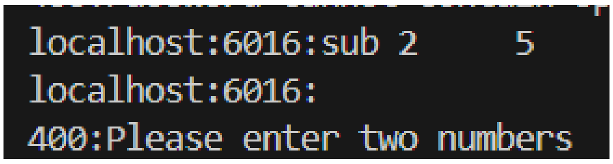
```
localhost:6016:sub 1
localhost:6016:
400:Please enter two numbers
localhost:6016:add 2
localhost:6016:
400:Invalid command
localhost:6016:changepwd
localhost:6016:
400:Please enter the new password
localhost:6016:changepwd a a
localhost:6016:
400:Password cannot contain spaces
```

Figure 11: Too many or too few arguments



```
localhost:6016:eat
localhost:6016:
400:Invalid command
```

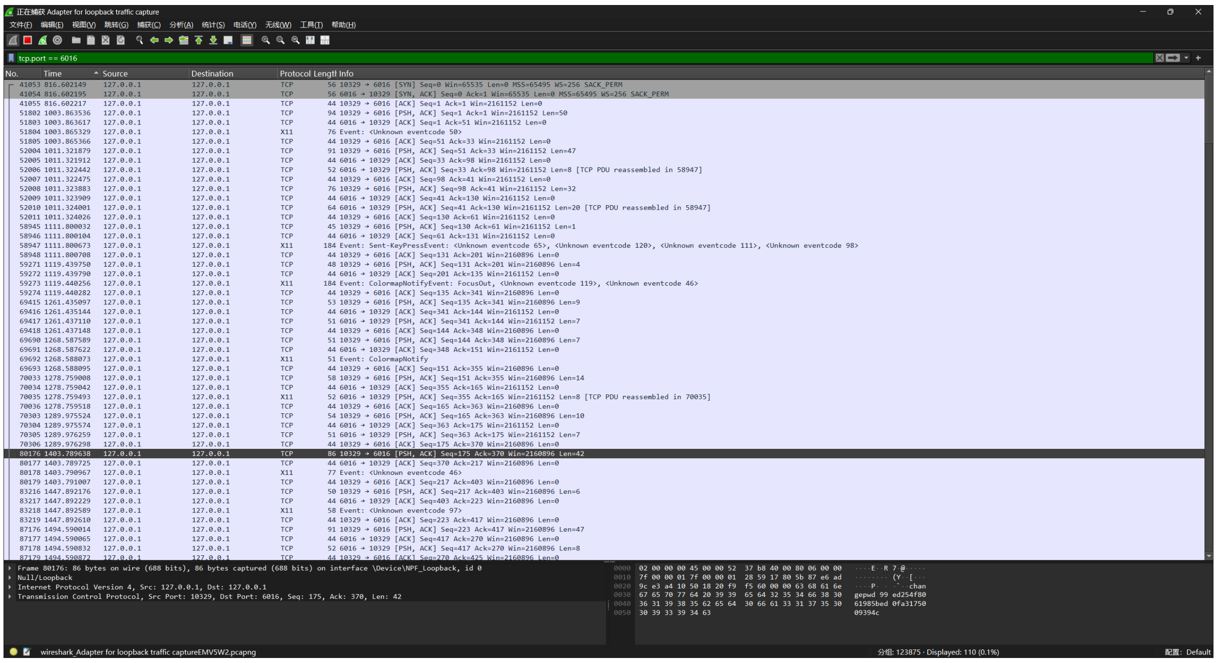
Figure 12: Command does not exist



```
localhost:6016:sub 2     5
localhost:6016:
400:Please enter two numbers
```

Figure 13: Extra space in the command

2 Wireshark capture



正在捕获 Adapter for loopback traffic capture
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

```
tcp.port == 6016
No.   Time          Source       Destination  Protocol Length Info
41053 816.602149    127.0.0.1    127.0.0.1    TCP      56 10329 → 6016 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
41054 816.602195    127.0.0.1    127.0.0.1    TCP      56 6016 → 10329 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
41055 816.602217    127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
51802 1003.863536   127.0.0.1    127.0.0.1    TCP      94 10329 → 6016 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=50
51803 1003.863617   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=1 Ack=51 Win=2161152 Len=0
51804 1003.865329   127.0.0.1    127.0.0.1    X11      76 Event: <Unknown eventcode 50>
51805 1003.865366   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=51 Ack=33 Win=2161152 Len=0
52004 1011.321879   127.0.0.1    127.0.0.1    TCP      91 10329 → 6016 [PSH, ACK] Seq=51 Ack=33 Win=2161152 Len=47
52005 1011.321912   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=33 Ack=98 Win=2161152 Len=0
52006 1011.322442   127.0.0.1    127.0.0.1    TCP      52 6016 → 10329 [PSH, ACK] Seq=33 Ack=98 Win=2161152 Len=8 [TCP PDU reassembled in 58947]
52007 1011.322475   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=98 Ack=41 Win=2161152 Len=0
52008 1011.323883   127.0.0.1    127.0.0.1    TCP      76 10329 → 6016 [PSH, ACK] Seq=98 Ack=41 Win=2161152 Len=32
52009 1011.323909   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=41 Ack=130 Win=2161152 Len=0
52010 1011.324001   127.0.0.1    127.0.0.1    TCP      64 6016 → 10329 [PSH, ACK] Seq=41 Ack=130 Win=2161152 Len=20 [TCP PDU reassembled in 58947]
52011 1011.324026   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=130 Ack=61 Win=2161152 Len=0
58945 1111.800032   127.0.0.1    127.0.0.1    TCP      45 10329 → 6016 [PSH, ACK] Seq=130 Ack=61 Win=2161152 Len=1
58946 1111.800104   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=61 Ack=131 Win=2161152 Len=0
58947 1111.800673   127.0.0.1    127.0.0.1    X11     184 Event: Sent-KeyPressEvent: <Unknown eventcode 65>, <Unknown eventcode 120>, <Unknown eventcode 111>, <Unknown eventcode 98>
58948 1111.800708   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=131 Ack=201 Win=2160896 Len=0
59271 1119.439750   127.0.0.1    127.0.0.1    TCP      48 10329 → 6016 [PSH, ACK] Seq=131 Ack=201 Win=2160896 Len=4
59272 1119.439790   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=201 Ack=135 Win=2161152 Len=0
59273 1119.440256   127.0.0.1    127.0.0.1    X11     184 Event: ColormapNotifyEvent: FocusOut, <Unknown eventcode 119>, <Unknown eventcode 46>
59274 1119.440282   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=135 Ack=341 Win=2160896 Len=0
69415 1261.435097   127.0.0.1    127.0.0.1    TCP      53 10329 → 6016 [PSH, ACK] Seq=135 Ack=341 Win=2160896 Len=9
69416 1261.435144   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=341 Ack=144 Win=2161152 Len=0
69417 1261.437110   127.0.0.1    127.0.0.1    TCP      51 6016 → 10329 [PSH, ACK] Seq=341 Ack=144 Win=2161152 Len=7
69418 1261.437148   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=144 Ack=348 Win=2160896 Len=0
69690 1268.587589   127.0.0.1    127.0.0.1    TCP      51 10329 → 6016 [PSH, ACK] Seq=144 Ack=348 Win=2160896 Len=7
69691 1268.587622   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=348 Ack=151 Win=2161152 Len=0
69692 1268.588073   127.0.0.1    127.0.0.1    X11      51 Event: ColormapNotify
69693 1268.588095   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=151 Ack=355 Win=2160896 Len=0
70033 1278.759008   127.0.0.1    127.0.0.1    TCP      58 10329 → 6016 [PSH, ACK] Seq=151 Ack=355 Win=2160896 Len=14
70034 1278.759042   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=355 Ack=165 Win=2161152 Len=0
70035 1278.759493   127.0.0.1    127.0.0.1    X11      52 6016 → 10329 [PSH, ACK] Seq=355 Ack=165 Win=2161152 Len=8 [TCP PDU reassembled in 70035]
70036 1278.759518   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=165 Ack=363 Win=2160896 Len=0
70303 1289.975524   127.0.0.1    127.0.0.1    TCP      54 10329 → 6016 [PSH, ACK] Seq=165 Ack=363 Win=2160896 Len=10
70304 1289.975574   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=363 Ack=175 Win=2161152 Len=0
70305 1289.976259   127.0.0.1    127.0.0.1    TCP      51 6016 → 10329 [PSH, ACK] Seq=363 Ack=175 Win=2161152 Len=7
70306 1289.976298   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=175 Ack=370 Win=2160896 Len=0
80176 1403.789638   127.0.0.1    127.0.0.1    TCP      86 10329 → 6016 [PSH, ACK] Seq=175 Ack=370 Win=2160896 Len=42
80177 1403.789725   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=370 Ack=217 Win=2160896 Len=0
80178 1403.790967   127.0.0.1    127.0.0.1    X11      77 Event: <Unknown eventcode 46>
80179 1403.791007   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=217 Ack=403 Win=2160896 Len=0
83216 1447.892176   127.0.0.1    127.0.0.1    TCP      50 10329 → 6016 [PSH, ACK] Seq=217 Ack=403 Win=2160896 Len=6
83217 1447.892229   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=403 Ack=223 Win=2160896 Len=0
83218 1447.892589   127.0.0.1    127.0.0.1    X11      58 Event: <Unknown eventcode 97>
83219 1447.892610   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=223 Ack=417 Win=2160896 Len=0
87176 1494.590014   127.0.0.1    127.0.0.1    TCP      91 10329 → 6016 [PSH, ACK] Seq=223 Ack=417 Win=2160896 Len=47
87177 1494.590065   127.0.0.1    127.0.0.1    TCP      44 6016 → 10329 [ACK] Seq=417 Ack=270 Win=2160896 Len=0
87178 1494.590832   127.0.0.1    127.0.0.1    TCP      52 6016 → 10329 [PSH, ACK] Seq=417 Ack=270 Win=2160896 Len=8
87179 1494.590872   127.0.0.1    127.0.0.1    TCP      44 10329 → 6016 [ACK] Seq=270 Ack=425 Win=2160896 Len=0

> Frame 80176: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 10329, Dst Port: 6016, Seq: 175, Ack: 370, Len: 42

0000  02 00 00 00 45 00 00 52  37 b8 40 00 80 06 00 00   ····E··R 7·@·····
0010  7f 00 00 01 7f 00 00 01  28 59 17 80 5b 87 e6 ad   ········ (Y··[···
0020  9c e3 a4 10 50 18 20 f9  f5 60 00 00 63 68 61 6e   ····P· · ·`··chan
0030  67 65 70 77 64 20 39 39  65 64 32 35 34 66 38 30   gepwd 99 ed254f80
0040  36 31 39 38 35 62 65 64  30 66 61 33 31 37 35 30   61985bed 0fa31750
0050  30 39 33 39 34 63                                  09394c

wireshark_Adapter for loopback traffic captureEMV5W2.pcapng    分组: 123875 · Displayed: 110 (0.1%)    配置: Default
```

Figure 14: Part of the packet captured during the communication between the server and the client

3 Code explanation

Most code follows the instruction provided in the comments. Only two parts requires further explanation.

3.1 User Command Logging

```python
def log_message(client_address, message, log_file="client_messages.txt"):
    with open(log_file, "a") as f:
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
        f.write(f"[{timestamp}] {client_address}: {message}\n")

def main_loop(socket_conn, client_address, login_user):
    try:
        receive_data = socket_conn.recv(1024).decode('utf-8').strip()
        if not receive_data:
            return False, None
        else:
            log_message(client_address, receive_data)

    # process the command
```

After receiving a message from the client, the server logs the message with the function `log_message`. All messages whether valid or invalid are logged, with the timestamp and the client address to distinguish them. We append each log to the file `client_messages.txt`. This file manually does not need to be created manually, as the append mode will create it if it does not exist.

3.2 Receiving challenge message

```
def generate_challenge():
    return os.urandom(8)

def server_response(server, password_hash):
    response = server.recv(1024)
    try:
        decoded_response = response.decode('utf-8')
        if decoded_response.startswith(('200:', '400:')):
            return response
        else:
            challenge_response = calculate_response(password_hash, response)
            server.send(challenge_response)
            return server.recv(1024)
    except UnicodeDecodeError:
        challenge_response = calculate_response(password_hash, response)
        server.send(challenge_response)
        return server.recv(1024)
```

The `urandom` function generates 8 random bytes, which is the challenge message. On the client side, we need to distinguish between the challenge message and a normal message. We do this by checking the response in two steps, corresponding to the two major differences between the challenge message and a normal message. Firstly, normal messages are encoded in UTF-8, while the challenge message is not. Secondly, normal messages start with either `200:` or `400:`, while the challenge message does not. Thus, we first try to decode the message with UTF-8, and then check if it starts with `200:` or `400:`. Even in the rare case that the randomly generated challenge message is a valid UTF-8 string, it will very likely not start with `200:` or `400:`.