Securing QR Codes

▪

Implementation of Data Integrity System of QR Codes by Applying Hash

Jaewook Lee
CheongShim International Academy
Jaeyu Kim
North London Collegiate School Jeju
Jeong Heum Na
North London Collegiate School Jeju
Yena Lee
Dublin High School

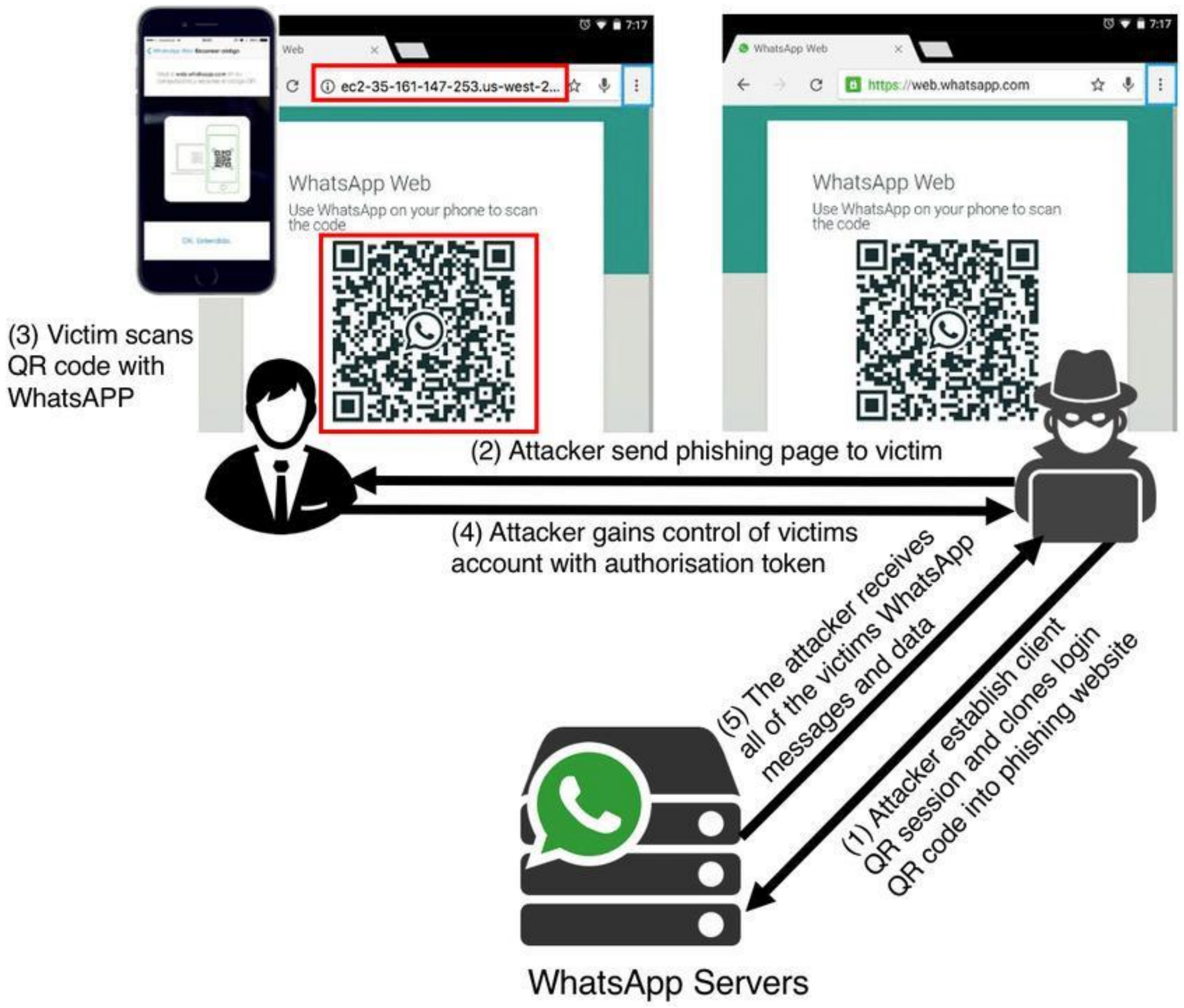
I. Abstract

The use of QR codes is growing as a convenient input mechanism to make mobile transactions more efficient. But Qshing, or the act of phishing for QR codes, has also been a growing cyber threat recently. Our innovative method is built on the foundation of identification, early intervention, and prevention through the use of the hash function. Using the hash function is the most effective prevention method to safeguard data integrity due to its unique ability to convert itself into a hash value of a short length with a simple algorithm. The use of MAC authentication method to QR code integrity as a working code should prevent hackers from gaining unapproved access to the data. The primary advantage of using the hash function to convert it into a hash value that is eventually used in conjunction with the MAC lies in the simplistic way of creating such code. Anyone with working knowledge of coding can create code protecting data integrity, and this form of prevention method can be applied to public facilities, banks, and shopping centers.

II. Introduction

A. What is Qshing? (QR Code Phishing)

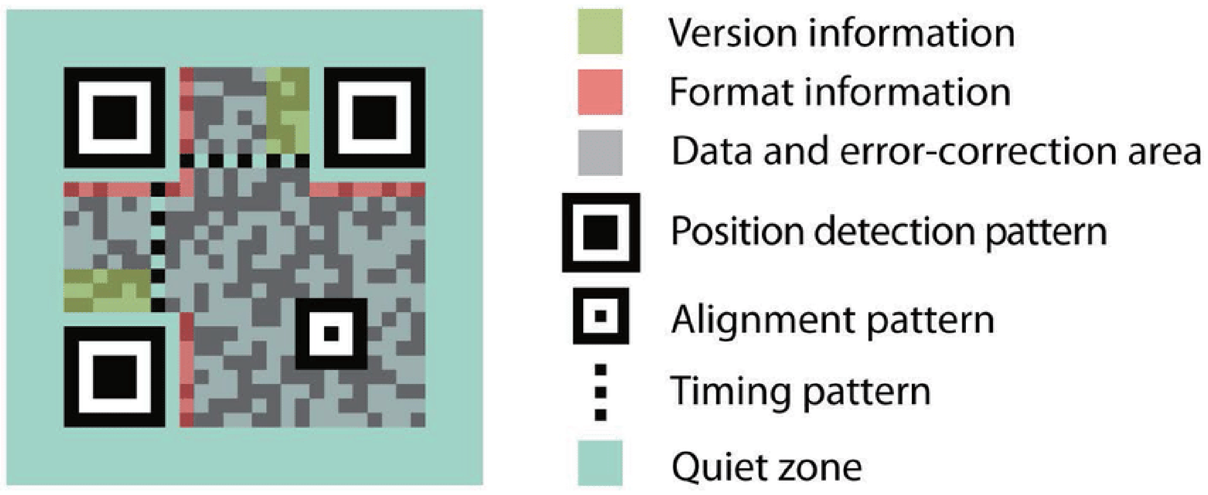
The novel term "Qshing" refers to a criminal method or activity that secretly replaces a genuine QR code with a modified or corrupted QR code and induces malicious programs to be installed unwittingly and steals personal information through the modified or corrupted QR code.


ec2-35-161-147-253.us-west-2...
https://web.whatsapp.com
WhatsApp Web
Use WhatsApp on your phone to scan the code
(3) Victim scans QR code with WhatsAPP
(2) Attacker send phishing page to victim
(4) Attacker gains control of victims account with authorisation token
(5) The attacker receives all of the victims WhatsApp messages and data
(1) Attacker establish client QR session and clones login QR code into phishing website
WhatsApp Servers

[Figure 1] WhatsApp QR link hijacking attack (initiated via sending a target victim a phishing website link). Here the spoofed QR code is automatically refreshed through the Javascript that has been copied from the legitimate WhatsApp authentication page. (Heartfield et al. 2018)

B. Qshing Using Site Phishing

The act of website phishing occurs when users are about to gain access to a particular Uniform Resource Locator (URL), colloquially termed "a web address," which is similar to the original website but in reality, a fake website. By inducing users to connect to a normal website that has been harmfully compromised from an already infected computer, hackers can successfully re-route unsuspecting users to the intended phishing website where hackers wait for users to reveal their credit cards information, social security numbers, biological information, and even sensitive passwords. By enticing users, this fake QR code can be used to inflict secondary damage that induces the installation of malicious applications not only on the PC but also on the victim's mobile phone.



[Figure 2] Structure of QR Code (Gao, Zhongpai et al. 2015)

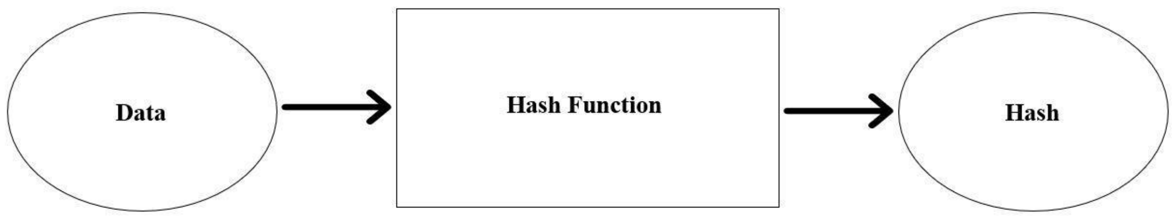
C. Qshing Using Man-In-The-Middle Attacks

"The Man-in-the-Middle Attack" is a cybersecurity attack technique through which a hacker infiltrates and manipulates data while two computers communicating via an open network. Each of these two computers is unwittingly connected to a hacker who infiltrates toward the middle ground where the hacker transmits the data transmitted from one computer to the other and successfully phishes a site or creates and replaces a genuine QR code with a fake one.

III. Objectives

Data integrity refers to an attribute that ensures that the original message or data has not been tampered with while distributing, transmitting, or storing data and personal files. In order to prevent phishing, integrity must be guaranteed. This study aims to ensure integrity in the process of QR code distribution.

A. Data Integrity: Hash



[Figure 3] Structure of Hash Function

B. Hash Characteristics

The hash function receives data of an arbitrary size as input and always outputs short and fixed length data of the same length. Since the hash function used in the security field additionally requires properties, such as reversed-phase resistance, second reversed-phase resistance, and collision resistance, there is an advantage that it is not possible to find out the original data by looking at the original hash value. The QR code integrity guarantee method proposed in this study prevents a Qshing attack by exploiting the fact that a Qshing attacker cannot find out the secret key used for MAC due to the following characteristics of the hash.

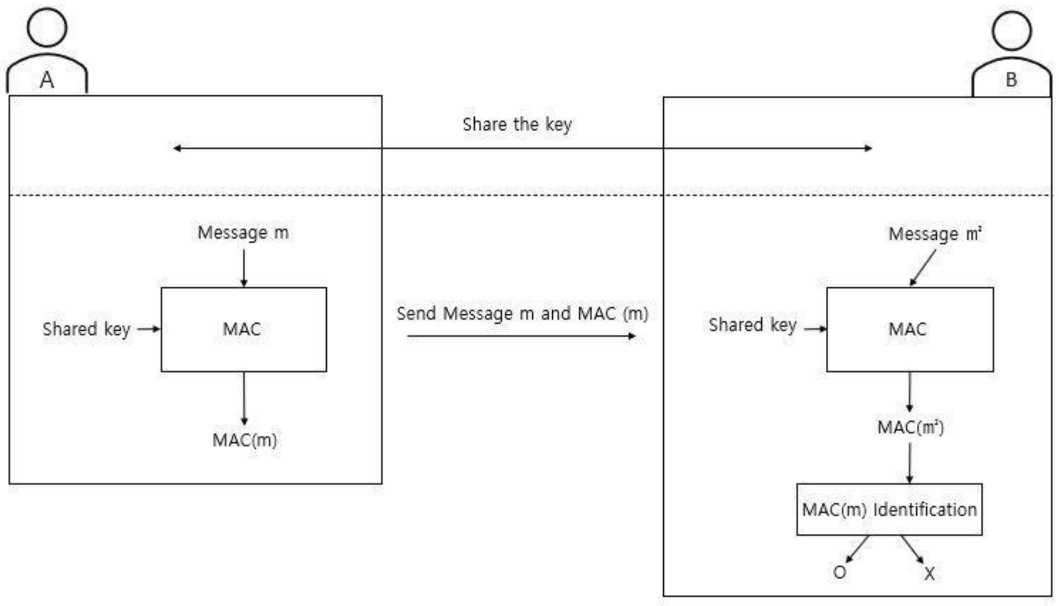
- Reverse Phase Resistance
  In the hash function, reversed-phase resistance means unidirectionality. When a user gives x as input to a hash function, look at the resulting y and a user should not be able to find x. When there is a hash function called h, it is a property that it should be computationally impossible to find x when the y value of y = h(x) is known.
- Second Reversed Phase Resistance
  In most cases, if the hash function h is used for data x, the result y=h(x) is unique. In this case, data x and x' having the same hash value are called "Second Inverse Phases", and the property of not finding a second inverse phase for the hash functions h and x is called "Second Inverse Resistance." To find the second inverse phase for a k-bit hash, the hash is a one-way

function, so there is only a method of making a hash 2^k times. This is practically impossible with the computational power of the current computer even if k is only about 80 bits.

- Collision Resistance
  Collision resistance is a property that makes it impossible to find a pair of x and x' where h(x)=h(x') and x != x' given a hash function h.

C. Data Integrity: MAC

MAC is short for Message Authentication Code, which a method of sending and receiving an authentication code with data so that the data receiver can check whether the data received from the sender is from the correct sender and that the data has not been forged.



[Figure 4] Procedure of Message Authentication Code (MAC)

First, sender A and receiver B share the secret key in advance. When sender A sends data to receiver B, it transmits the data and the MAC made using the data and the secret key together. Recipient B who received the data creates a MAC utilizing the data it received and the secret key shared in advance and compares it with the MAC received from sender A to see if it was sent by the sender A.

D. HMAC

In this study, HMAC, which is efficient for speed and computational processing using hashes, was used. The HMAC generation process is as follows.

(1) The sender divides the message into blocks having a size of b bits.

(2) If the size of the secret key k shared between the sender/receiver is smaller than b, pad it with 0 in front to match it with b bits.

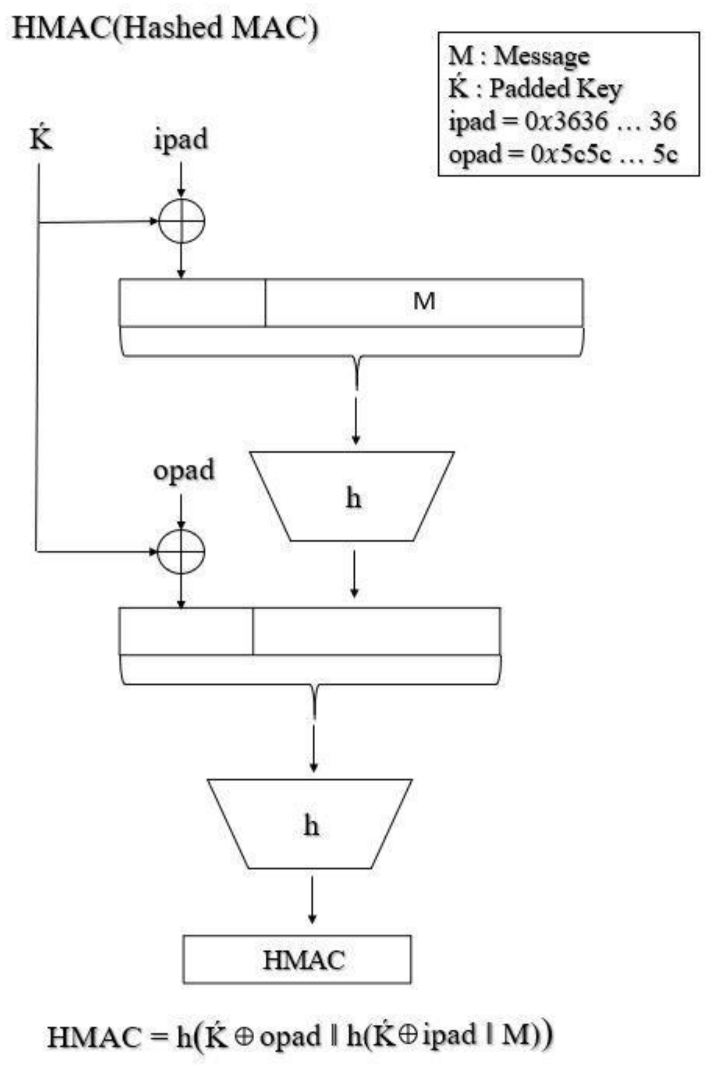
(3) The padded secret key k' performs XOR operation with a specific constant, ipad (Input Pad).

(4) The result from (3) is attached to the front of the message, and the hash value is calculated.

(5) As in (3), the padded secret key k' is XORed with opad (output pad) and then pasted in front of the hash value obtained in (4).

(6) Calculate the final hash value using the hash function used in (4).

If the sender delivers the hash value finally obtained in (6) with the message to the receiver, the receiver calculates the hash value through the same process as above using the previously shared secret key and the received message, and then checks whether it matches the received hash value. do. If this matches, it can be assumed can be seen that there was no forgery and alteration, and the integrity was guaranteed.



[Figure 5] HMAC Generation Process

IV. Methodology

A. QR Integrity Verification Program Algorithm

1) Definition of QR Integrity Verification Program Algorithm

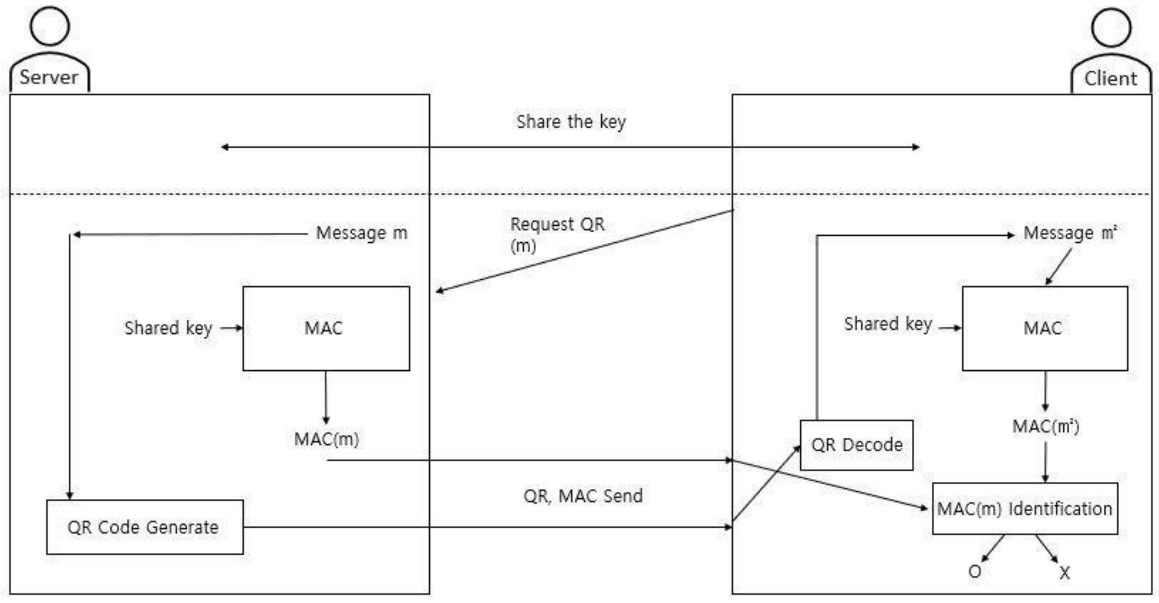
Typically, when a sender delivers a QR code to a recipient, it generates a QR code using the information to be transmitted and then delivers only the QR code to the recipient. To prevent forgery, this study designed a QR code integrity verification algorithm. This algorithm operates in the local environment of the sender and receiver, which is completely separate from the network that they send and receive information from when the sender is the sender, and the receiver uses the QR code. The QR code transmission process using this algorithm is as follows.

(1) The sender generates a QR code that contains the information to be provided to the receiver and generates a MAC using the information contained in it and a secret key shared with the receiver in advance.

(2) The sender delivers the results generated in (1) to the receiver.

(3) The receiver decodes the received QR code and calculates the MAC' using the information and secret key.

(4) The recipient compares the received MAC with the generated MAC', and if they are the same, the integrity of the QR code can be guaranteed.



[Figure 6] QR Integrity Verification Program Algorithm Chart

In this algorithm, in (1) and (4), the process of calculating and comparing the MAC by the sender and the receiver, respectively, is performed locally without using the network, so the attacker cannot obtain information on the secret key generated by the MAC.

2) QR Integrity Verification Program Algorithm Advantages

Even if a hacker intercepts the HMAC and QR code transmitted by the sender, the corresponding plaintext cannot be derived due to the characteristics of the cryptographic hash. As a result, the secret key cannot be found. Lastly, this algorithm is P2P authentication, which eliminates the intervention of a third party such as an accredited certification authority, so it is efficient in terms of time and processing cost.

B. QR Integrity Verification Program Algorithm Implementation

Followings are codes used for QR Integrity Verification Program Algorithm – qr_check.py

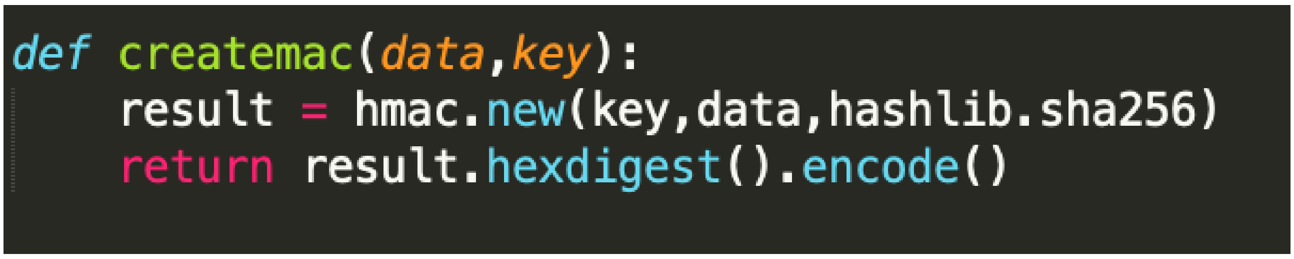
- Import Module

Below table is the description of the import modules used for qr_check.py.

Description	Module used for activating Hmac	Module used for activating Hash functions such as Md5, sha1

[Table 1] Descriptions for Import Modules

- Createmac

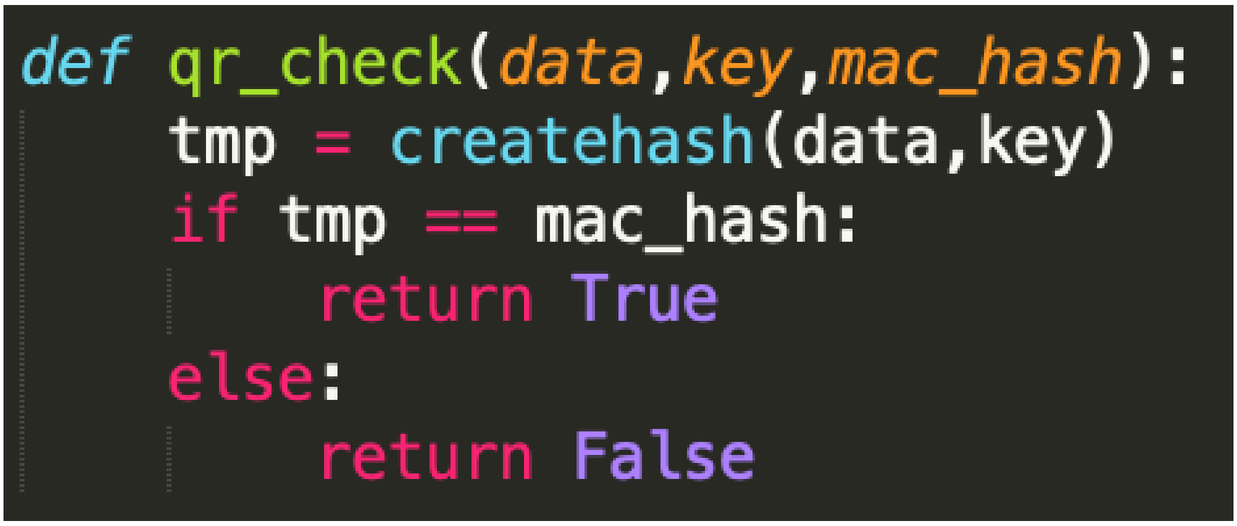


```python
def createmac(data,key):
    result = hmac.new(key,data,hashlib.sha256)
    return result.hexdigest().encode()
```

[Figure 7] Codes of Createmac

This is the code that creates the MAC. Receive data and secret key to be used for MAC generation as arguments. The hash used at this time is sha256, and other types of cryptographic hashes can be used.

- qr_check



```
def qr_check(data,key,mac_hash):
    tmp = createhash(data,key)
    if tmp == mac_hash:
        return True
    else:
        return False
```

[Figure 8] Codes of gr_check

This code receives data, secret key, and mac as arguments, and checks whether the mac created with the data and the secret key matches the mac received as an argument.

V. Design of Invention

A simple trial service was implemented to test whether the algorithm proposed in this study and the code prevented Qshing. For the Qshing test, a general server (127.0.0.1/9999) and a phishing server (127.0.0.1/9998) were produced.

A. Server Codes

When the client connects, this server identifies the client through register or login, generates a QR code with the data requested by the client, and sends it to the client.

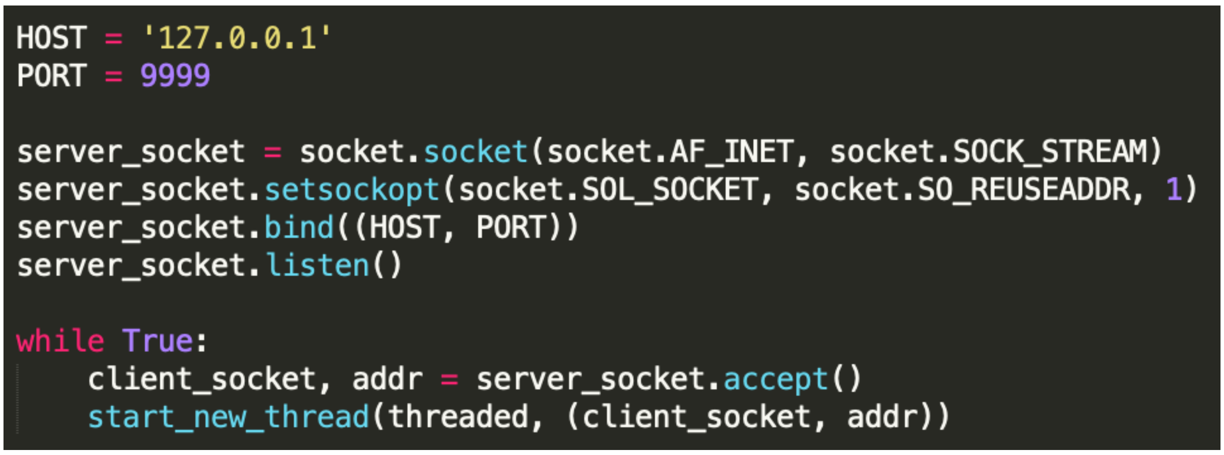
- Import Module

  Below table is description of the import modules used for server codes.

Name	Socket	QR Code	_Thread
Description	Module for socket transaction	Module for producing QR Code	Threads for transacting multiple clients

[Table 2] Modules of Server Codes

- Connect



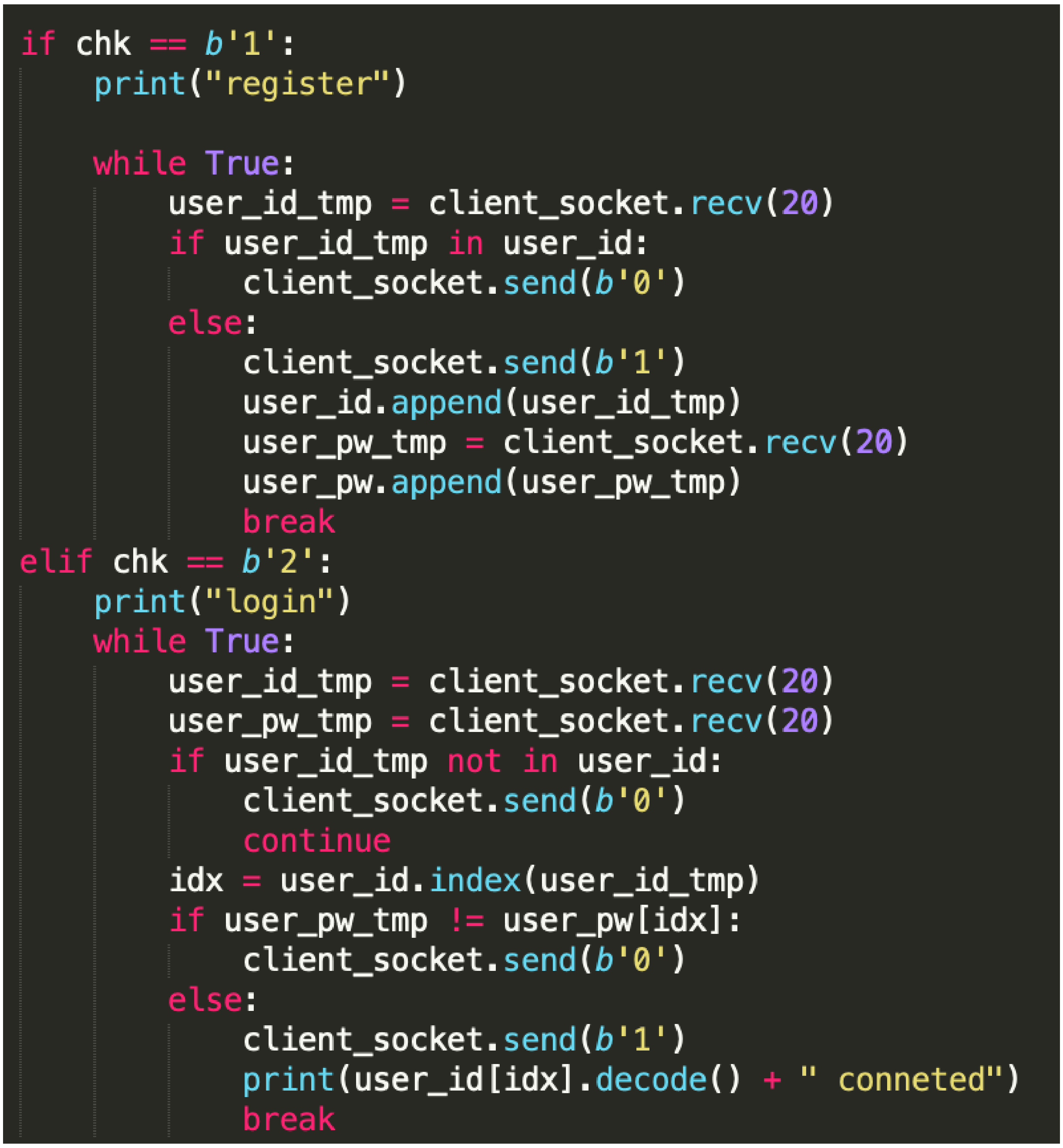
```
HOST = '127.0.0.1'
PORT = 9999

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((HOST, PORT))
server_socket.listen()

while True:
    client_socket, addr = server_socket.accept()
    start_new_thread(threaded, (client_socket, addr))
```

[Figure 9] Connecting Server Codes

After specifying the server's IP address and port, socket communication is prepared through the socket module, and when a client connects, it acts as a server using tread. This server uses 127.0.0.1 IP and 9999 port.

- Register / Login

This is a code part for register and login to identify the client. It delivers the ID duplicate check and login success to the client and stores the index of the client by classifying it using the ID of the connected client.

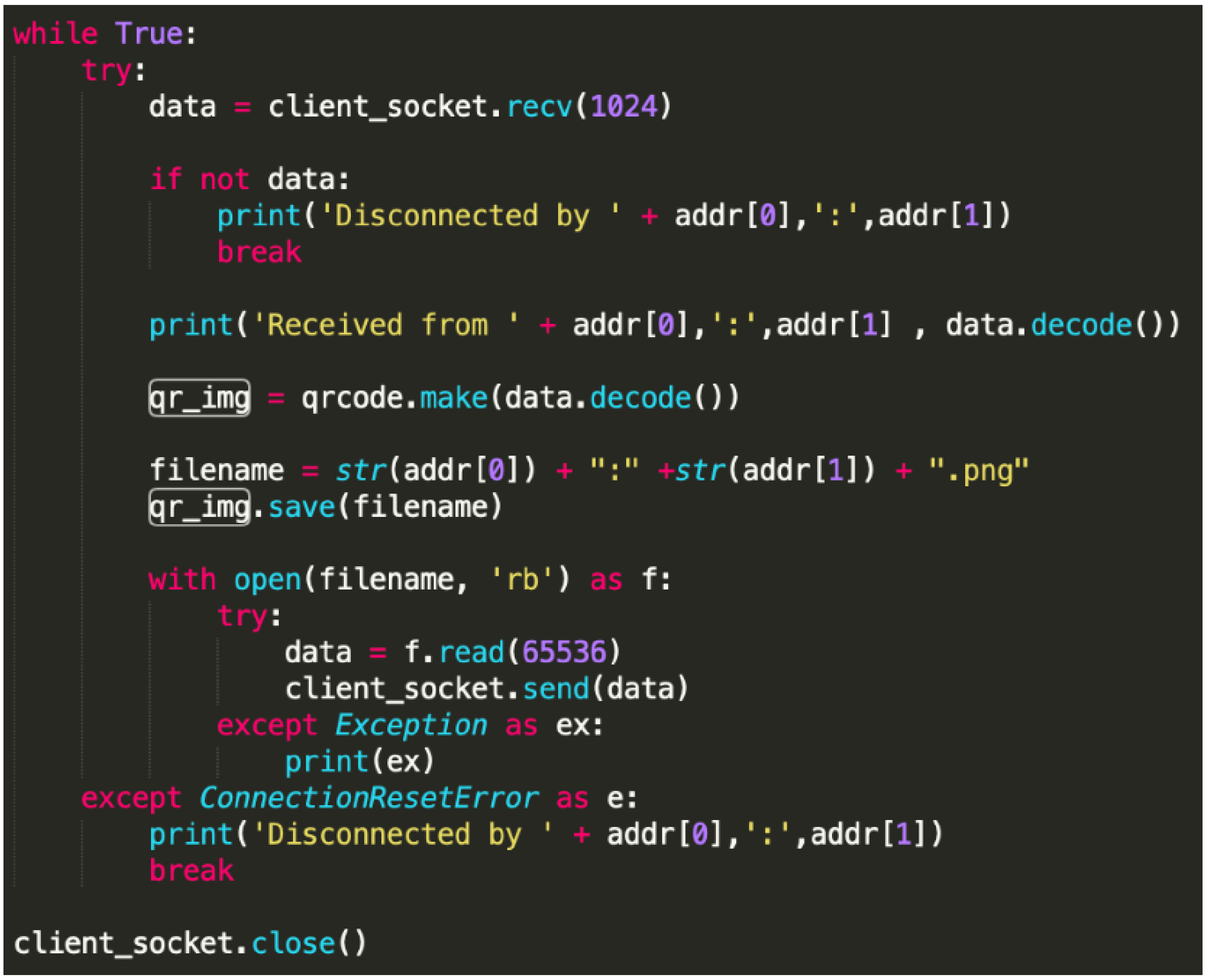


```python
if chk == b'1':
    print("register")

    while True:
        user_id_tmp = client_socket.recv(20)
        if user_id_tmp in user_id:
            client_socket.send(b'0')
        else:
            client_socket.send(b'1')
            user_id.append(user_id_tmp)
            user_pw_tmp = client_socket.recv(20)
            user_pw.append(user_pw_tmp)
            break
elif chk == b'2':
    print("login")
    while True:
        user_id_tmp = client_socket.recv(20)
        user_pw_tmp = client_socket.recv(20)
        if user_id_tmp not in user_id:
            client_socket.send(b'0')
            continue
        idx = user_id.index(user_id_tmp)
        if user_pw_tmp != user_pw[idx]:
            client_socket.send(b'0')
        else:
            client_socket.send(b'1')
            print(user_id[idx].decode() + " conneted")
            break
```

[Figure 10] Code Part for Register/Login

- Qr Code Generation and Transmission

The QR code is generated using the qrcode module, and the QR code is transmitted through socket communication.



```
while True:
    try:
        data = client_socket.recv(1024)

        if not data:
            print('Disconnected by ' + addr[0],':',addr[1])
            break

        print('Received from ' + addr[0],':',addr[1] , data.decode())

        qr_img = qrcode.make(data.decode())

        filename = str(addr[0]) + ":" +str(addr[1]) + ".png"
        qr_img.save(filename)

        with open(filename, 'rb') as f:
            try:
                data = f.read(65536)
                client_socket.send(data)
            except Exception as ex:
                print(ex)
    except ConnectionResetError as e:
        print('Disconnected by ' + addr[0],':',addr[1])
        break

client_socket.close()
```

[Figure 11] Code Part for Generating and Transmitting a QR code

B. Client Code Description

The client connects to the server, registers, or login, sends the data that it wants to create with QR to the server, receives the QR code from the server, and stores it in the current directory.

- Import Modules

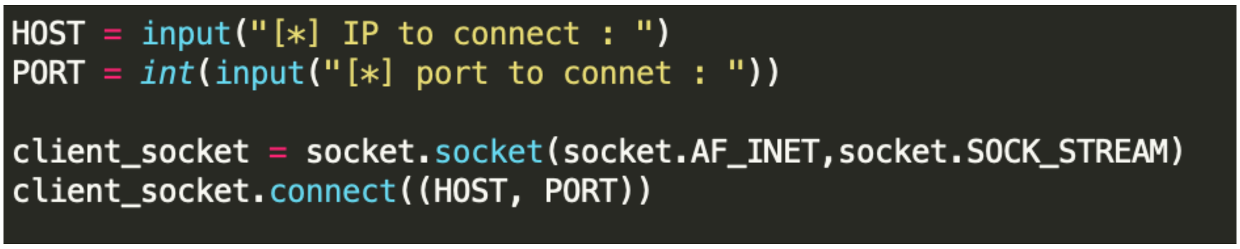
Below table is a description of the import modules used for client codes.

Name	Socket	PIL
Description	Module for socket transaction	Module for visualizing QR codes

[Table 3] Modules of Client Codes

- Connect

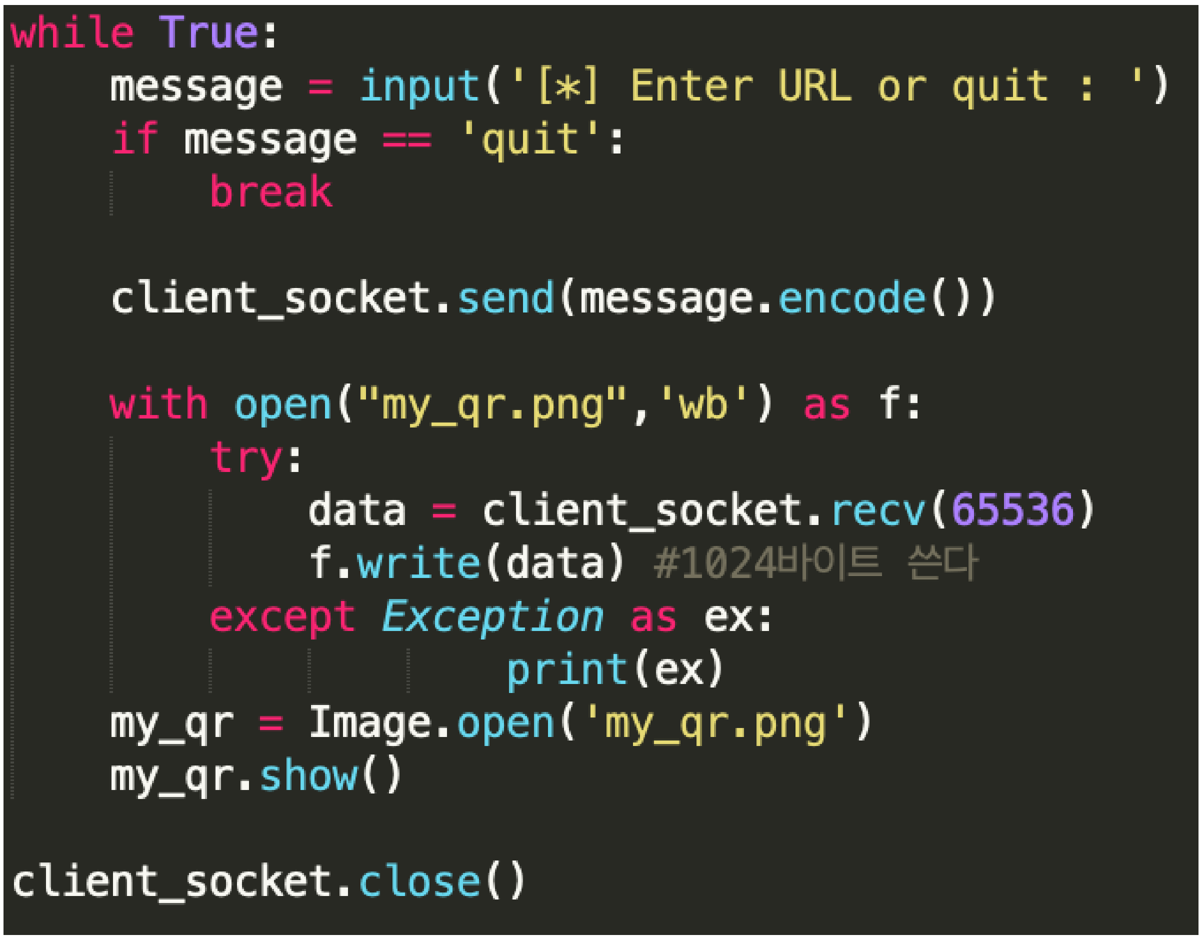
It is a code part for connection with the server and connects to the corresponding address by inputting IP and port.



```
HOST = input("[*] IP to connect : ")
PORT = int(input("[*] port to connet : "))

client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
client_socket.connect((HOST, PORT))
```

[Figure 12] Code Parts for Connection with the Server

- Data transmission, QR reception

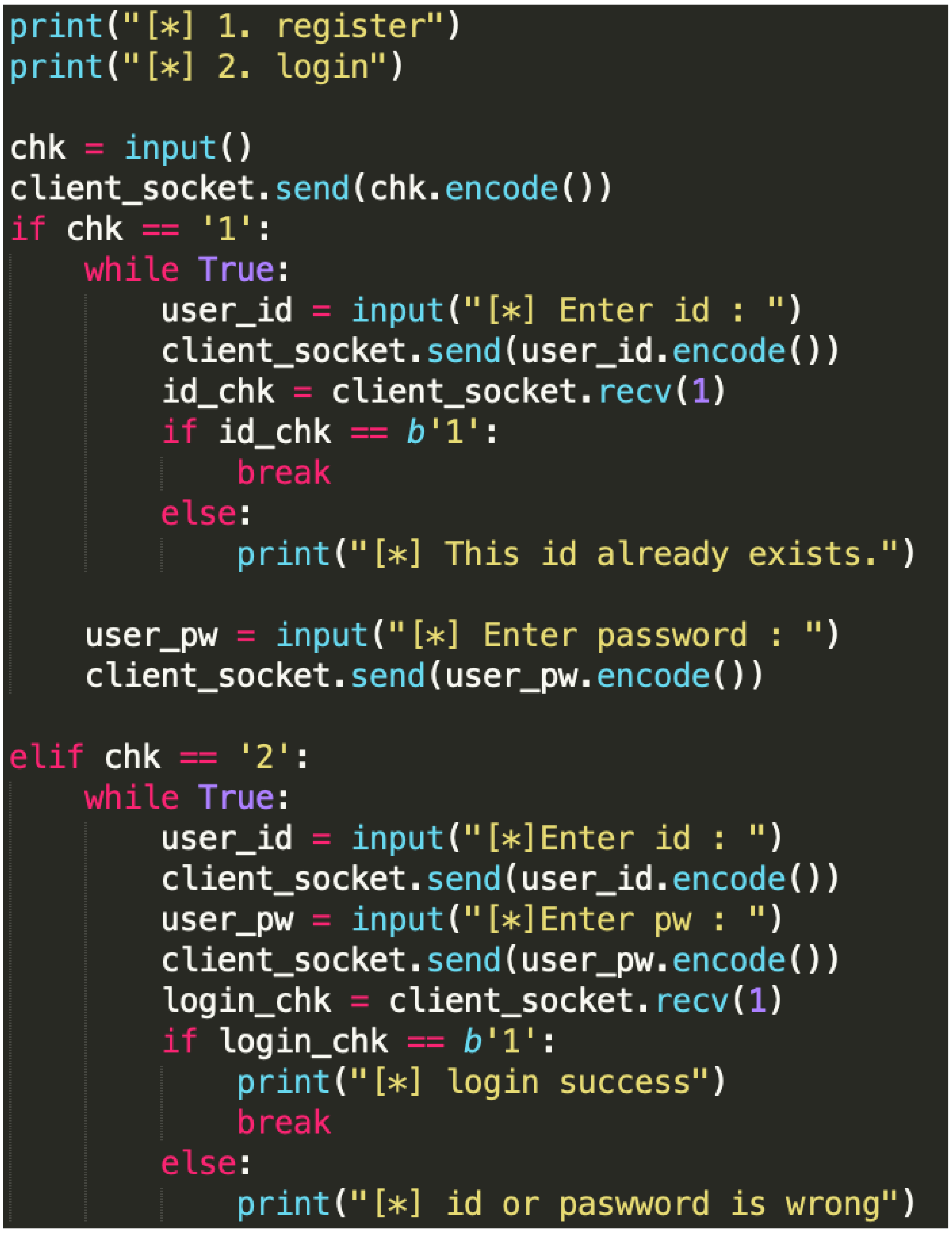


```python
while True:
    message = input('[*] Enter URL or quit : ')
    if message == 'quit':
        break

    client_socket.send(message.encode())

    with open("my_qr.png",'wb') as f:
        try:
            data = client_socket.recv(65536)
            f.write(data) #1024바이트 쓴다
        except Exception as ex:
                print(ex)
    my_qr = Image.open('my_qr.png')
    my_qr.show()

client_socket.close()
```

[Figure 13] Codes for Data Transmission and QR reception

The corresponding QR code is received and displayed to the user through the PIL module, and this QR code is saved in the current directory.

- Register, Login



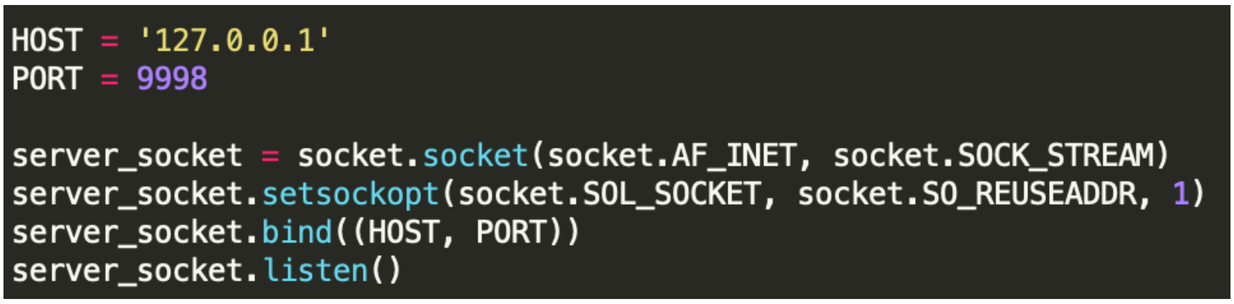
```python
print("[*] 1. register")
print("[*] 2. login")

chk = input()
client_socket.send(chk.encode())
if chk == '1':
    while True:
        user_id = input("[*] Enter id : ")
        client_socket.send(user_id.encode())
        id_chk = client_socket.recv(1)
        if id_chk == b'1':
            break
        else:
            print("[*] This id already exists.")

    user_pw = input("[*] Enter password : ")
    client_socket.send(user_pw.encode())

elif chk == '2':
    while True:
        user_id = input("[*]Enter id : ")
        client_socket.send(user_id.encode())
        user_pw = input("[*]Enter pw : ")
        client_socket.send(user_pw.encode())
        login_chk = client_socket.recv(1)
        if login_chk == b'1':
            print("[*] login success")
            break
        else:
            print("[*] id or pasword is wrong")
```

[Figure 14] Code Part for Register and Login

This is a code part for register and login. If it is successful, the service can be started by receiving whether the ID is duplicated or whether the login is successful.

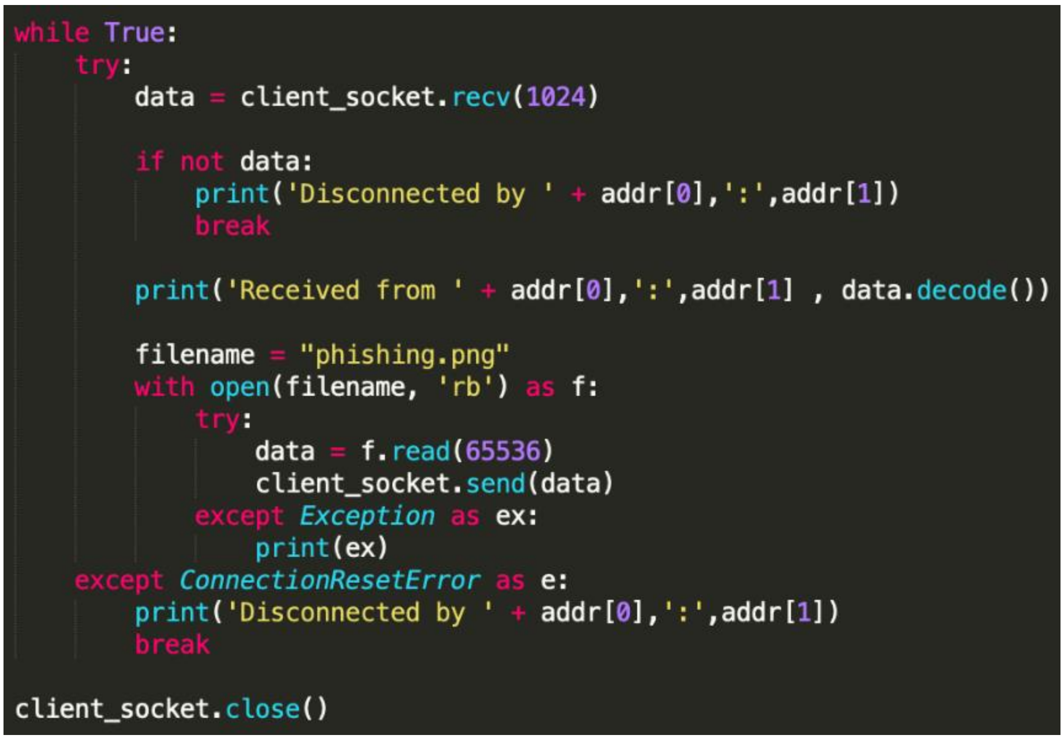
- Phishing Server Code Description

Phishing server is a server for testing in which a client is being phished. Unlike a normal server, it uses port 9998 instead of 9999, and always connects to the service regardless of the client's id/pw.



```
HOST = '127.0.0.1'
PORT = 9998

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((HOST, PORT))
server_socket.listen()
```

[Figure 15] Code for Phishing Sever



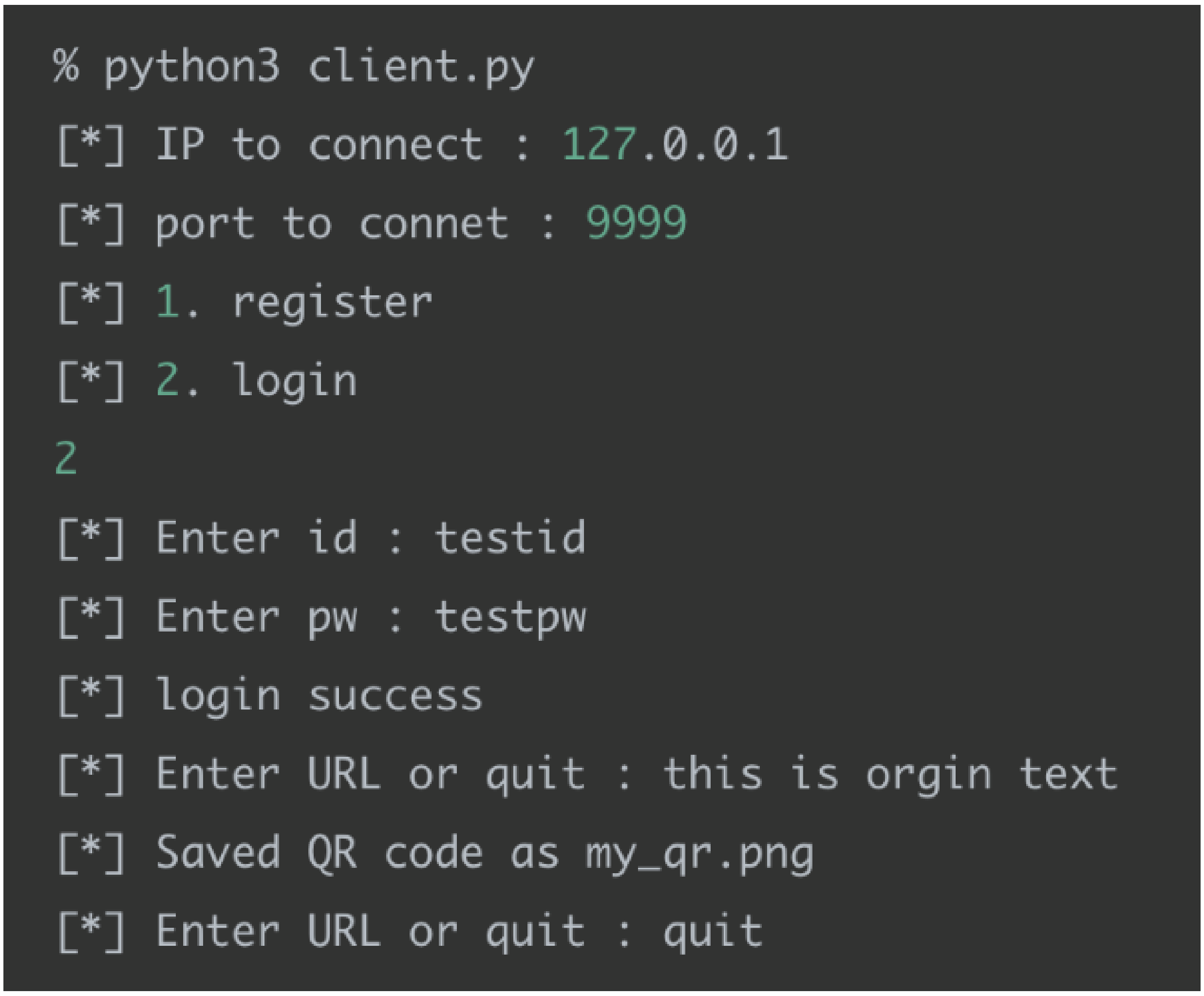
```
while True:
    try:
        data = client_socket.recv(1024)

        if not data:
            print('Disconnected by ' + addr[0],':',addr[1])
            break

        print('Received from ' + addr[0],':',addr[1] , data.decode())

        filename = "phishing.png"
        with open(filename, 'rb') as f:
            try:
                data = f.read(65536)
                client_socket.send(data)
            except Exception as ex:
                print(ex)
    except ConnectionResetError as e:
        print('Disconnected by ' + addr[0],':',addr[1])
        break

client_socket.close()
```

[Figure 16] Phishing Server Sending Malicious QR Code

VI. Result

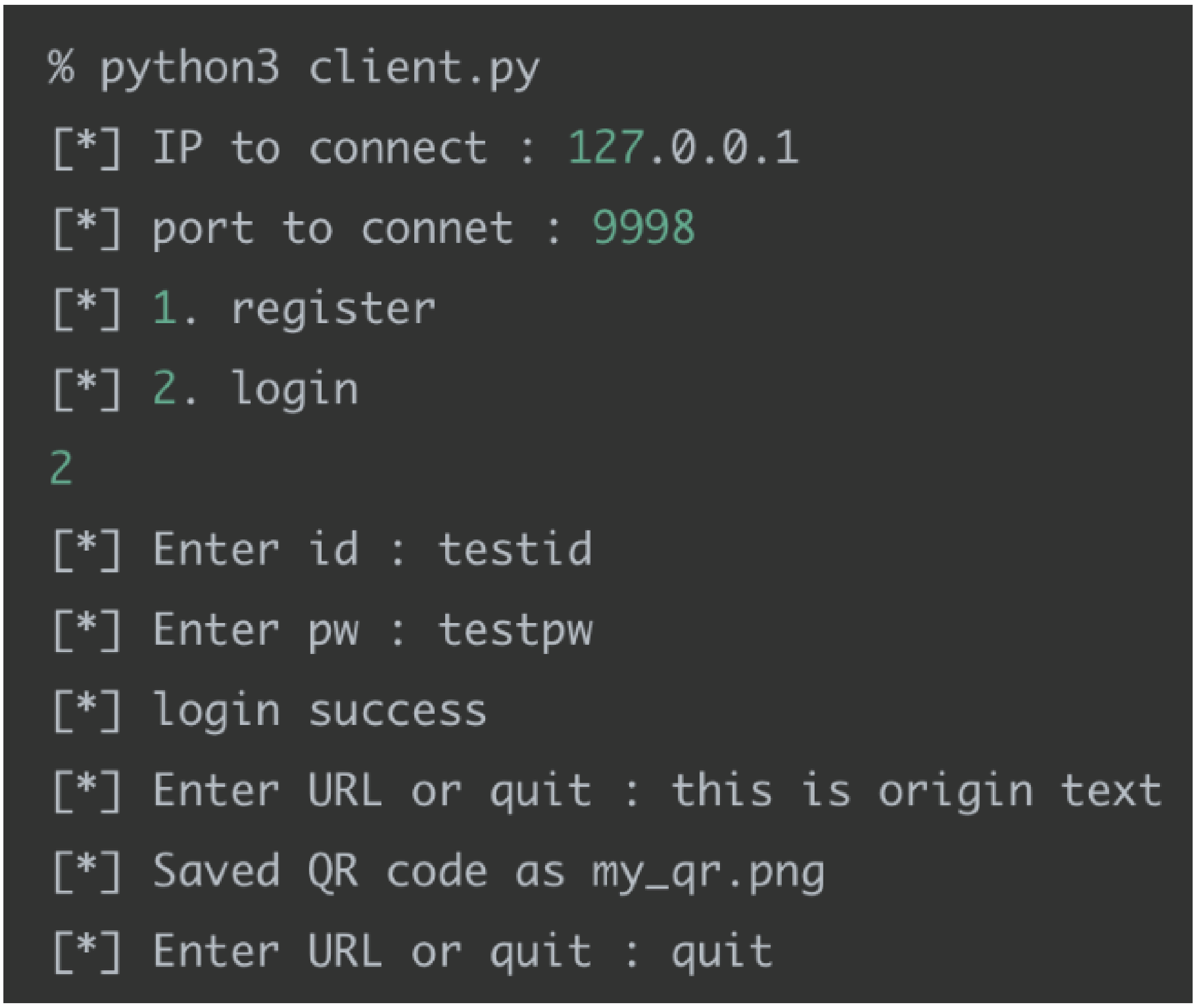
A. Results Before Implementation of QR Integrity Verification Program Algorithm

The result of executing the previously written codes is as follows:



```
% python3 client.py
[*] IP to connect : 127.0.0.1
[*] port to connet : 9999
[*] 1. register
[*] 2. login
2
[*] Enter id : testid
[*] Enter pw : testpw
[*] login success
[*] Enter URL or quit : this is orgin text
[*] Saved QR code as my_qr.png
[*] Enter URL or quit : quit
```

[Figure 17] Terminal result window where a client connects to a general server before implementing QR Integrity Verification Program Algorithm



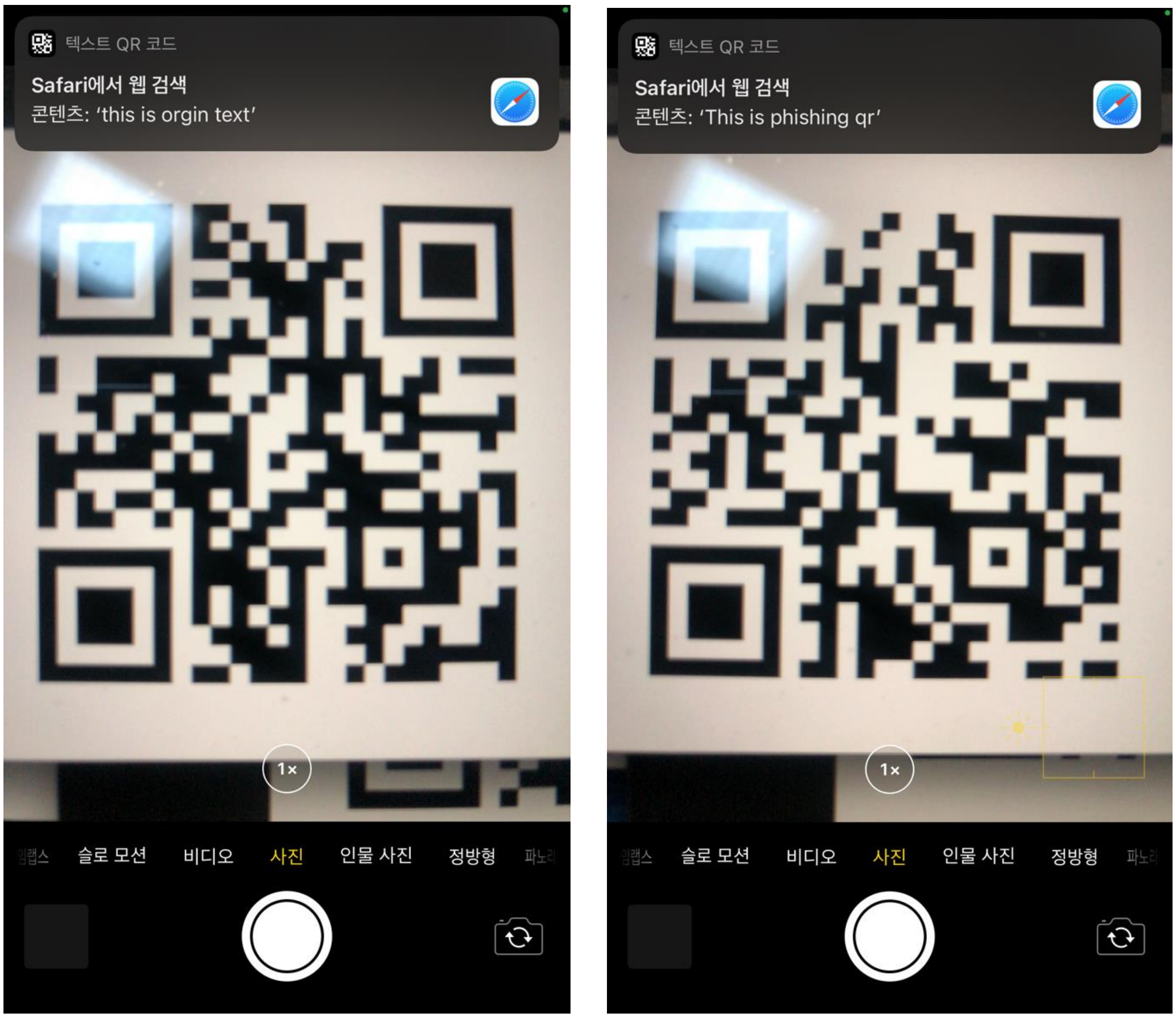
```
% python3 client.py
[*] IP to connect : 127.0.0.1
[*] port to connet : 9998
[*] 1. register
[*] 2. login
2
[*] Enter id : testid
[*] Enter pw : testpw
[*] login success
[*] Enter URL or quit : this is origin text
[*] Saved QR code as my_qr.png
[*] Enter URL or quit : quit
```

[Figure 18] Result window when a client connects to a phishing server before implementing QR Integrity Verification Program Algorithm

[Figure 17] is a terminal result window where a client connects to a general server, logs in with test ID, requests a QR code for data called this is origin text, receives it, and saves it. And **[Figure 18]** is a result window when a client accidentally connects to a phishing server pretending to provide the same service as a general server and performs the same action.

Both result windows show the same result that the same QR code of my_qr.png is saved. Thus, it is **NOT** known whether the client has connected to the phishing server or not.

If one scans the QR code received from each server with a mobile phone, one can see that different data is derived as shown in the picture above. **[Figure 17]** is a QR code received from a general server, which contains the data requested by the client as it is. On the other hand, **[Figure 18]** is a QR code received from a phishing server, and it can be seen that phishing data is contained regardless of the data requested by the client.

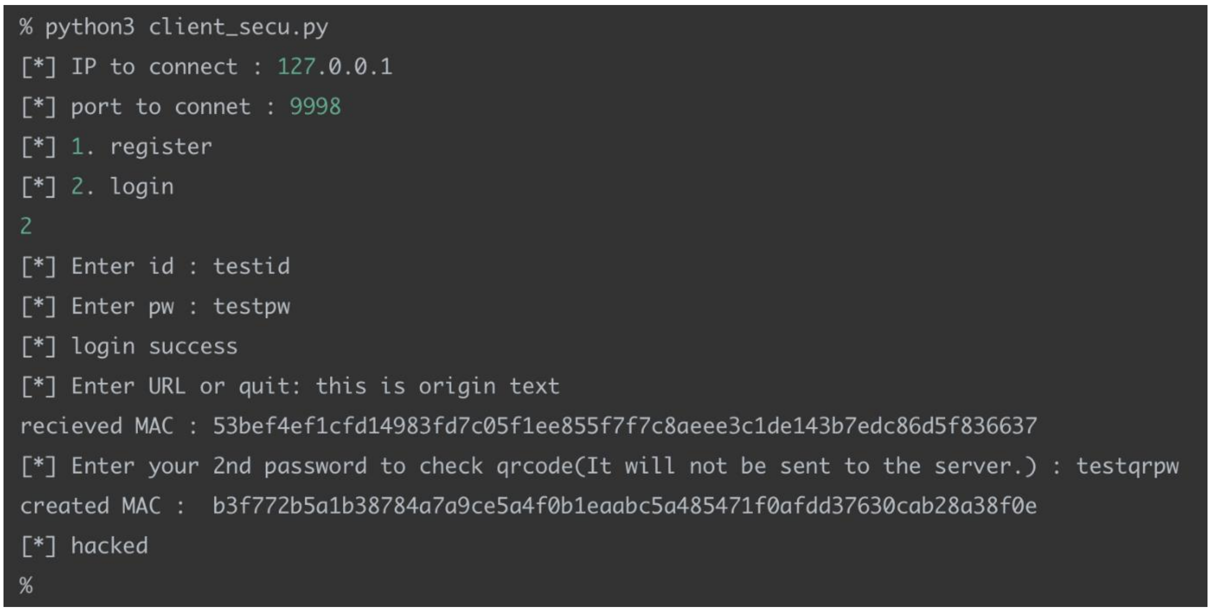

텍스트 QR 코드
Safari에서 웹 검색
콘텐츠: 'this is orgin text'

텍스트 QR 코드
Safari에서 웹 검색
콘텐츠: 'This is phishing qr'

[Figure 19] QR scanning shows two different outcomes through our QR Integrity Verification Program Algorithm

B. Results After the Implementation of QR Integrity Verification Program Algorithm

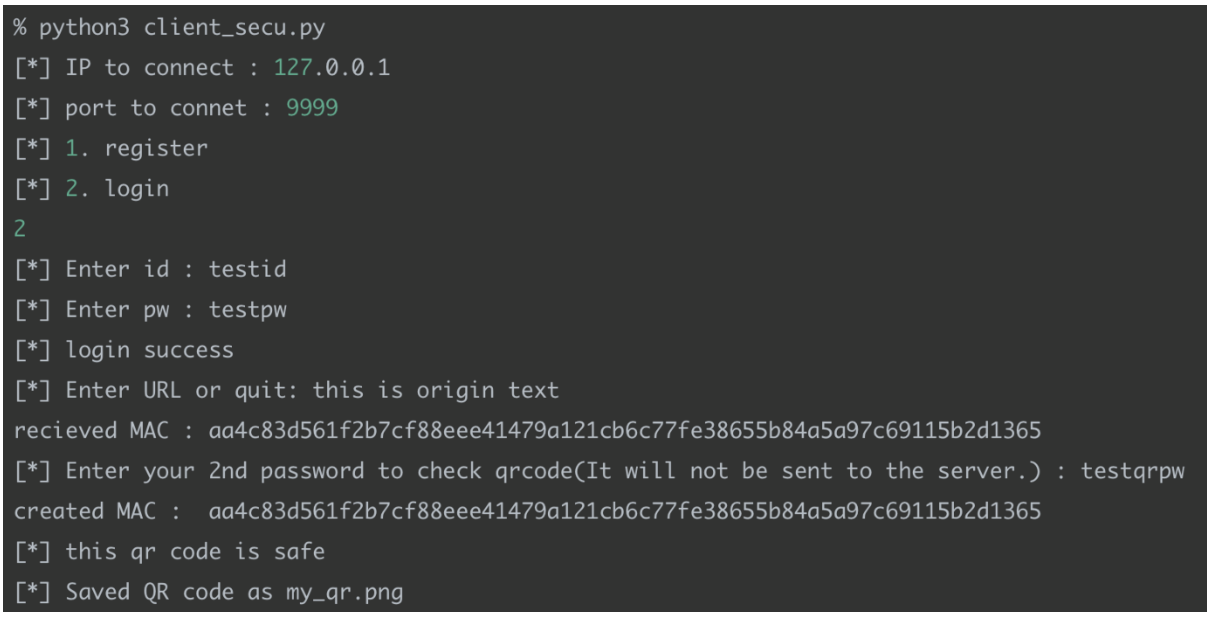
This time, using qr_check.py written in 4.1 as a module, we look at the results of applying the QR code integrity verification algorithm suggested in this study to the server and client. The difference from the above general server/client is that the private key for MAC generation is set during the registration process, and the corresponding secret key is used during the QR code transmission process and QR code verification process. This is achieved by adding the createmac function and qr_check function of qr_check.py to the existing code only a little.



```
% python3 client_secu.py
[*] IP to connect : 127.0.0.1
[*] port to connet : 9998
[*] 1. register
[*] 2. login
2
[*] Enter id : testid
[*] Enter pw : testpw
[*] login success
[*] Enter URL or quit: this is origin text
recieved MAC : 53bef4ef1cfd14983fd7c05f1ee855f7f7c8aeee3c1de143b7edc86d5f836637
[*] Enter your 2nd password to check qrcode(It will not be sent to the server.) : testqrpw
created MAC :  b3f772b5a1b38784a7a9ce5a4f0b1eaabc5a485471f0afdd37630cab28a38f0e
[*] hacked
%
```

[Figure 20] Result window when a client connects to a phishing server after implementing QR Integrity Verification Program Algorithm

Users who were introduced to the QR code integrity verification method presented in this study can still distinguish whether or not they are visiting Qshing sites through the MAC authentication, even if they receive a malicious QR code from a phishing server as shown in the **[Figure 18]**. The received MAC and the received QR code and the MAC created using the private key do not match, and the program is automatically terminated with the text hacked after recognizing that it is Qshing.



```
% python3 client_secu.py
[*] IP to connect : 127.0.0.1
[*] port to connet : 9999
[*] 1. register
[*] 2. login
2
[*] Enter id : testid
[*] Enter pw : testpw
[*] login success
[*] Enter URL or quit: this is origin text
recieved MAC : aa4c83d561f2b7cf88eee41479a121cb6c77fe38655b84a5a97c69115b2d1365
[*] Enter your 2nd password to check qrcode(It will not be sent to the server.) : testqrpw
created MAC :  aa4c83d561f2b7cf88eee41479a121cb6c77fe38655b84a5a97c69115b2d1365
[*] this qr code is safe
[*] Saved QR code as my_qr.png
```

[Figure 21] Terminal result window where a client connects to a general server after implementing QR Integrity Verification Program Algorithm

The program confirms that the two MACs match, prints a text stating that the integrity of the QR code is verified and safe, and saves the QR code.

VII. Conclusion

This paper was able to present and implement a QR code integrity verification system that is practically effective in preventing Qshing. (QR Code Phishing) Since the QR code integrity verification process is based on security using the characteristics of the cryptographic hash function, the safety itself is secured. Even if the QR image is intercepted and altered, it is impossible to create a legitimate MAC because the secret key is unknown.

As for potential development, the MAC is inserted into the QR image using steganography. As a result,, it is possible to verify the integrity by sending and receiving only the QR code image without having to send the MAC value separately as in this study.

VIII. References

Gao, Zhongpai & Zhai, Guangtao & Hu, Chunjia. (2015). The Invisible QR Code. 10.1145/2733373.2806398.

Heartfield, Ryan & Loukas, George. (2018). Protection Against Semantic Social Engineering Attacks. 10.1007/978-3-319-97643-3_4.