Roy Soldin 204542179
Eran Levav 203394465
Shir Cohen 313233074

Topics in Network Secutiry

Encrypted CLI Email Client with Antivirus

Tool Objective:

In this project we chose to implement a CLI Email client with antivirus and encryption capabilities. The client can send and receive messages through Gmail account using SMTP and IMAP protocols. Our client Email can receive messages only from addresses which compose a list of contacts, when receiving a message from non-contact the user will choose if to accept or decline the message. In case a message is to be sent or received with an attachment, prior to the action's invocation (sending or saving to the disk), the client will run it through the extensive analyzation of Virus Total. In addition our service also allows sending and receiving encrypted messages using Caesar Cipher encryption. The user who receives an encrypted message can only read the content of the message using our tool and not via Gmail normal inbox.

Project Challenges

Our main challenge when working on this project was to combine user friendly tool with affective security capabilities, in order to achieve this goal we had to use the right Python packages to get an easy to use CLI menu and all the antivirus options we wanted to have. In addition, another challenge was to write our project in a manner that will be easy to read and understand, we use multi classes style of code to enable later expanding our tool capabilities and in the same time make the code more readable.

Weakness and Strength:

Strength:

- Implementing the tool in a modular manner for later expanding and adding extra features.
- Easy to use CLI menu for sending and receiving messages.
- Permission to receive messages only from a list of contacts with the option to accept a message from unknown address.
- Option for encrypt and decrypt messages.

Weakness

- While not logged in to our service, the Email box can receive messages from unknown addresses without getting blocked.
- Reading encrypted messages only from our CLI prompt and not from the regular Gmail inbox.

Description of The Service work and components

Our tool is composed from 3 main classes:

1. SMTP : responsible for everything that involve sending a message which includes getting all the relevant data from the user (address, subject, body, files, etc.), scanning the attached files and encrypt the body of the message if the user chose to.
2. IMAP : responsible for fetching all messages from the Email inbox to the user, while scanning all files for viruses and decrypt if necessary.
3. Security Email System : in charge of all user interface.

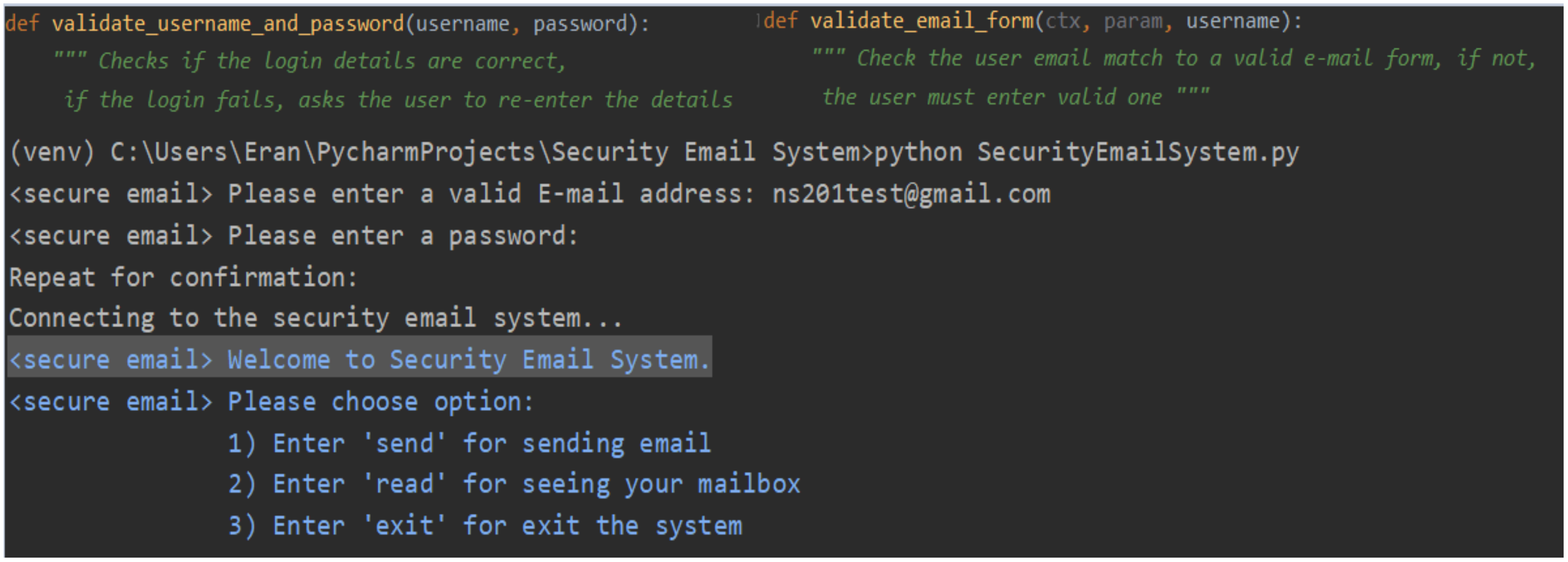
In order to work with the tool, the user must enter a valid Gmail credentials, after a successful login the user will choose an option from the menu – either to send a message, receive a message or exit the system.

1. If the user chose to send a message the system will ask to provide all information needed and will give the user an option to encrypt the message.
2. If the user chose to receive, the system will fetch all the mails from the user's inbox and if one of the messages is not from a contact the system will notify that to the user and give him an option to decline (delete the message) or accept the message.

Full documentation in the code.

Proof of Concept

Scenario 1 - Logging to the service



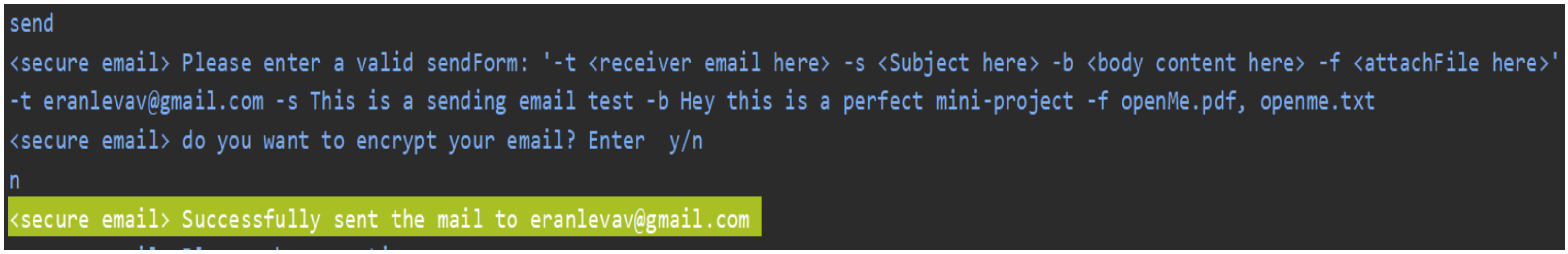
```
def validate_username_and_password(username, password):
    """ Checks if the login details are correct,
     if the login fails, asks the user to re-enter the details
```

```
def validate_email_form(ctx, param, username):
    """ Check the user email match to a valid e-mail form, if not,
     the user must enter valid one """
```

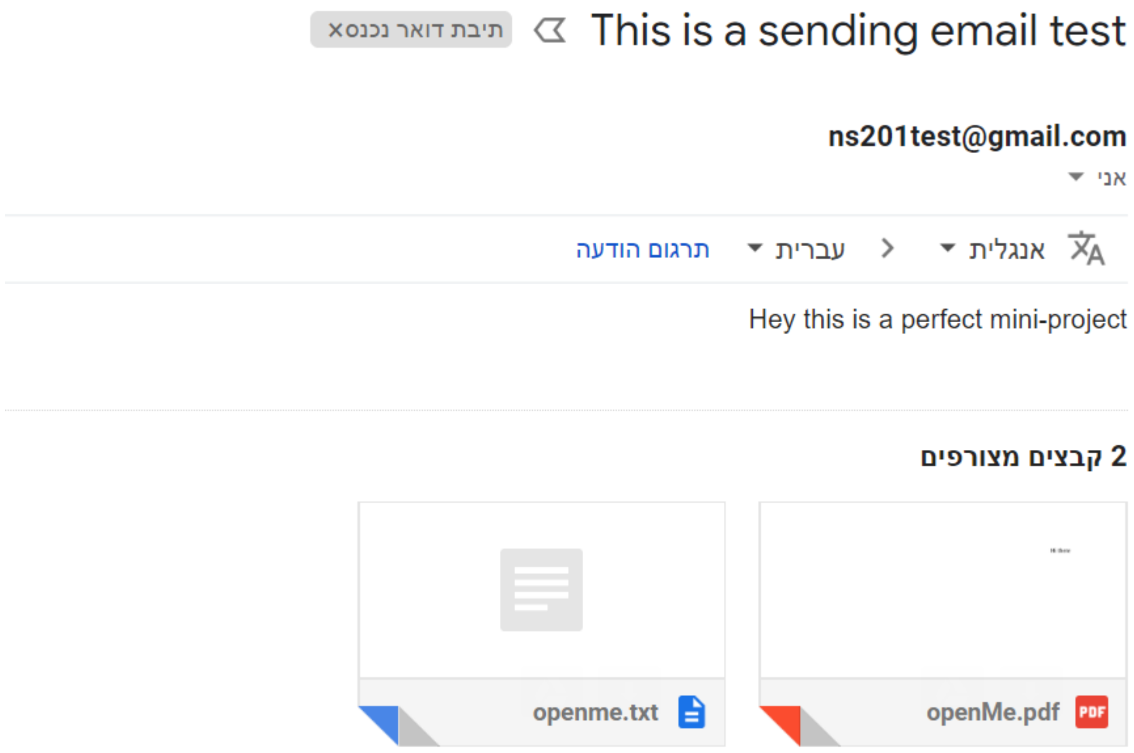
```
(venv) C:\Users\Eran\PycharmProjects\Security Email System>python SecurityEmailSystem.py
<secure email> Please enter a valid E-mail address: ns201test@gmail.com
<secure email> Please enter a password:
Repeat for confirmation:
Connecting to the security email system...
<secure email> Welcome to Security Email System.
<secure email> Please choose option:
               1) Enter 'send' for sending email
               2) Enter 'read' for seeing your mailbox
               3) Enter 'exit' for exit the system
```

When logging in, the system validate the email address form and only then try to connect to Gmail service, typing the password is hidden for security measures. Once connected, the system prompts the main menu.

Scenario 2 - Sending a message



```
send
<secure email> Please enter a valid sendForm: '-t <receiver email here> -s <Subject here> -b <body content here> -f <attachFile here>'
-t eranlevav@gmail.com -s This is a sending email test -b Hey this is a perfect mini-project -f openMe.pdf, openme.txt
<secure email> do you want to encrypt your email? Enter  y/n
n
<secure email> Successfully sent the mail to eranlevav@gmail.com
```



This is a sending email test

תיבת דואר נכנס ×

ns201test@gmail.com

אני

אנגלית > עברית תרגום הודעה

Hey this is a perfect mini-project

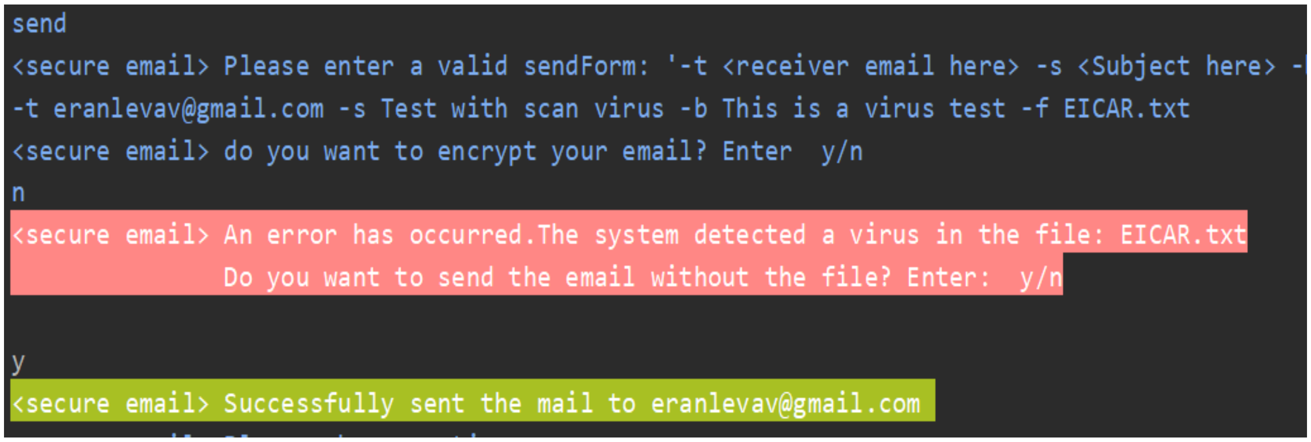
2 קבצים מצורפים

openMe.pdf

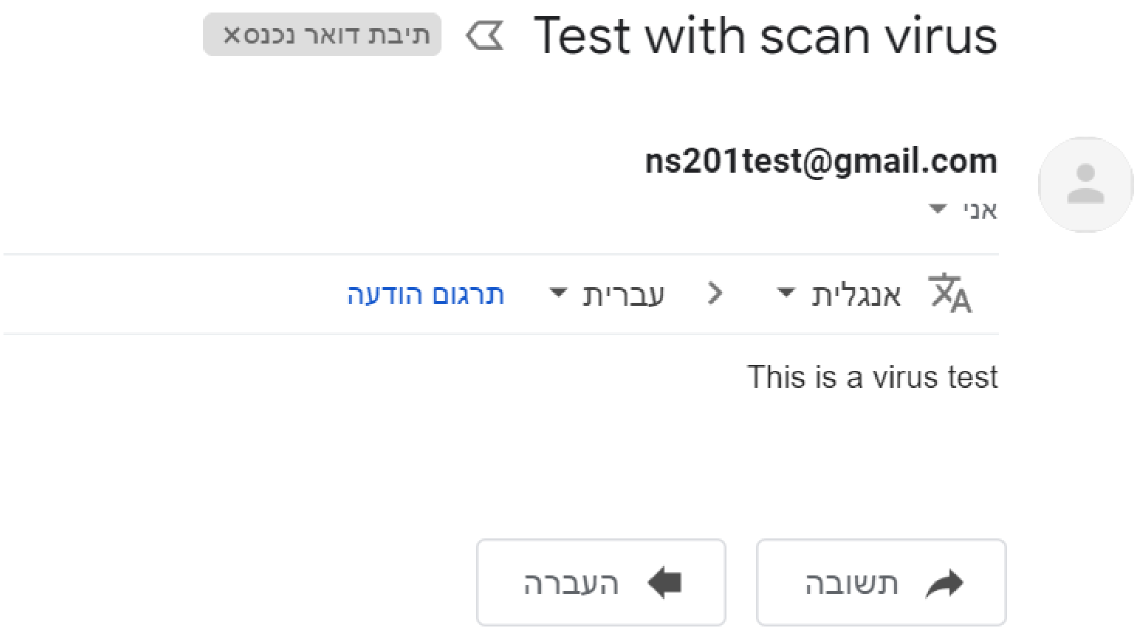
openme.txt

When the user want to send a message, the system shows the template required. As shown in picture, a message with attachment was successfully sent.

Scenario 3 - Sending a message with malicious file



```
send
<secure email> Please enter a valid sendForm: '-t <receiver email here> -s <Subject here> -
-t eranlevav@gmail.com -s Test with scan virus -b This is a virus test -f EICAR.txt
<secure email> do you want to encrypt your email? Enter  y/n
n
<secure email> An error has occurred.The system detected a virus in the file: EICAR.txt
               Do you want to send the email without the file? Enter:  y/n

y
<secure email> Successfully sent the mail to eranlevav@gmail.com
```



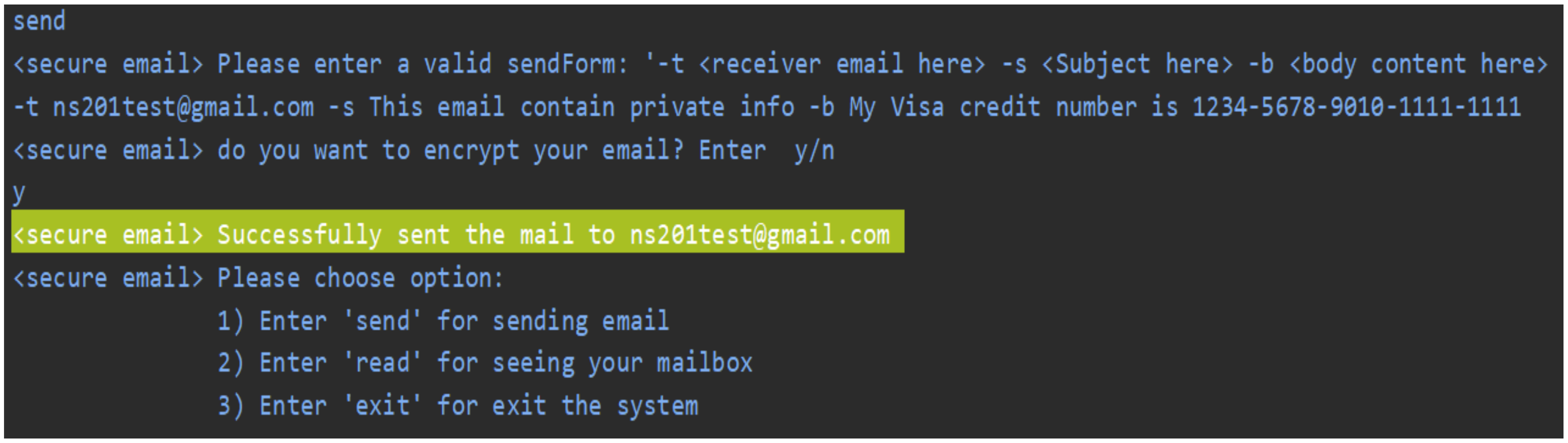
Test with scan virus

ns201test@gmail.com

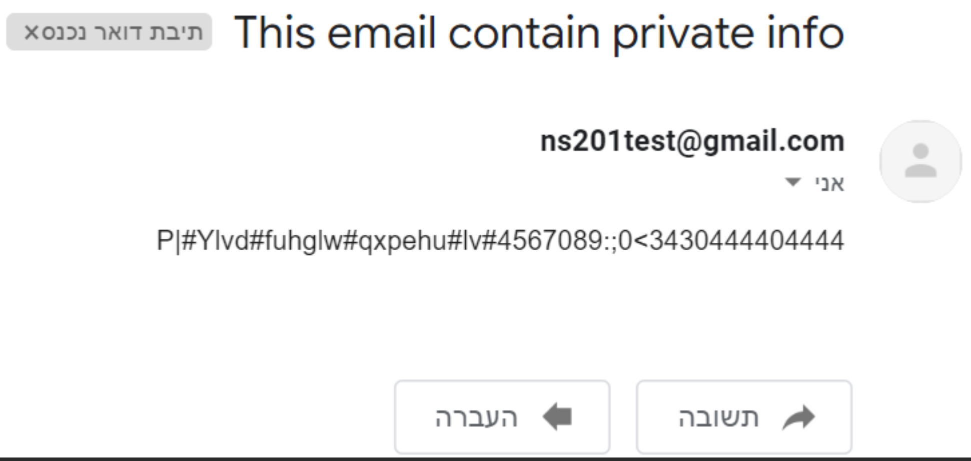
This is a virus test

The system detects that a file with malware is trying to be sent. Then the system asks the user whether to send the message without the malicious file or to abort the action (similar to Gmail behavior).

Scenario 4 - Sending a message with encrypted private information



```
send
<secure email> Please enter a valid sendForm: '-t <receiver email here> -s <Subject here> -b <body content here>
-t ns201test@gmail.com -s This email contain private info -b My Visa credit number is 1234-5678-9010-1111-1111
<secure email> do you want to encrypt your email? Enter  y/n
y
<secure email> Successfully sent the mail to ns201test@gmail.com
<secure email> Please choose option:
               1) Enter 'send' for sending email
               2) Enter 'read' for seeing your mailbox
               3) Enter 'exit' for exit the system
```

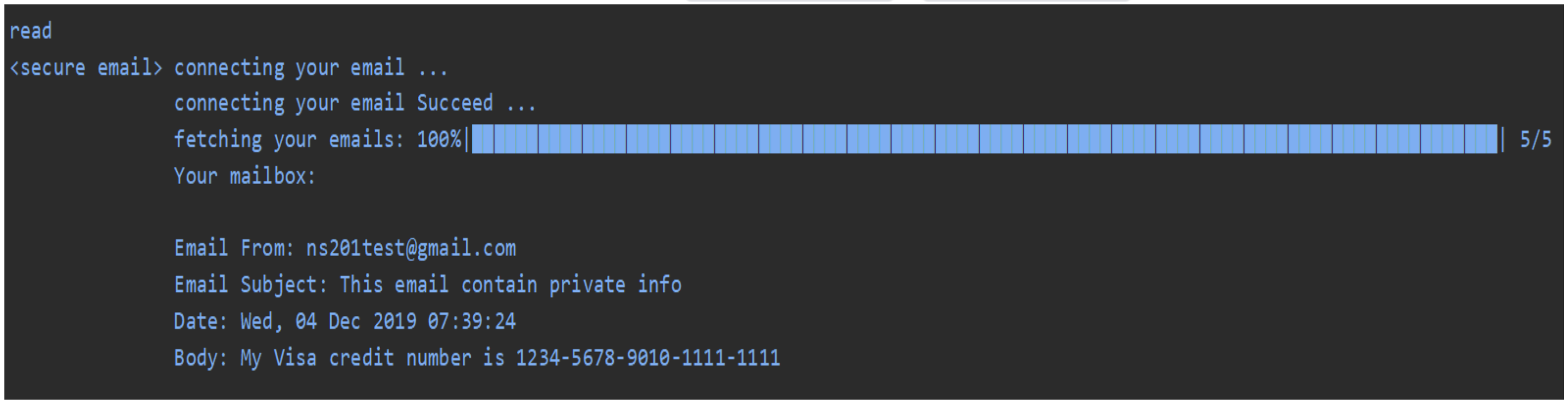


This email contain private info

ns201test@gmail.com

אני

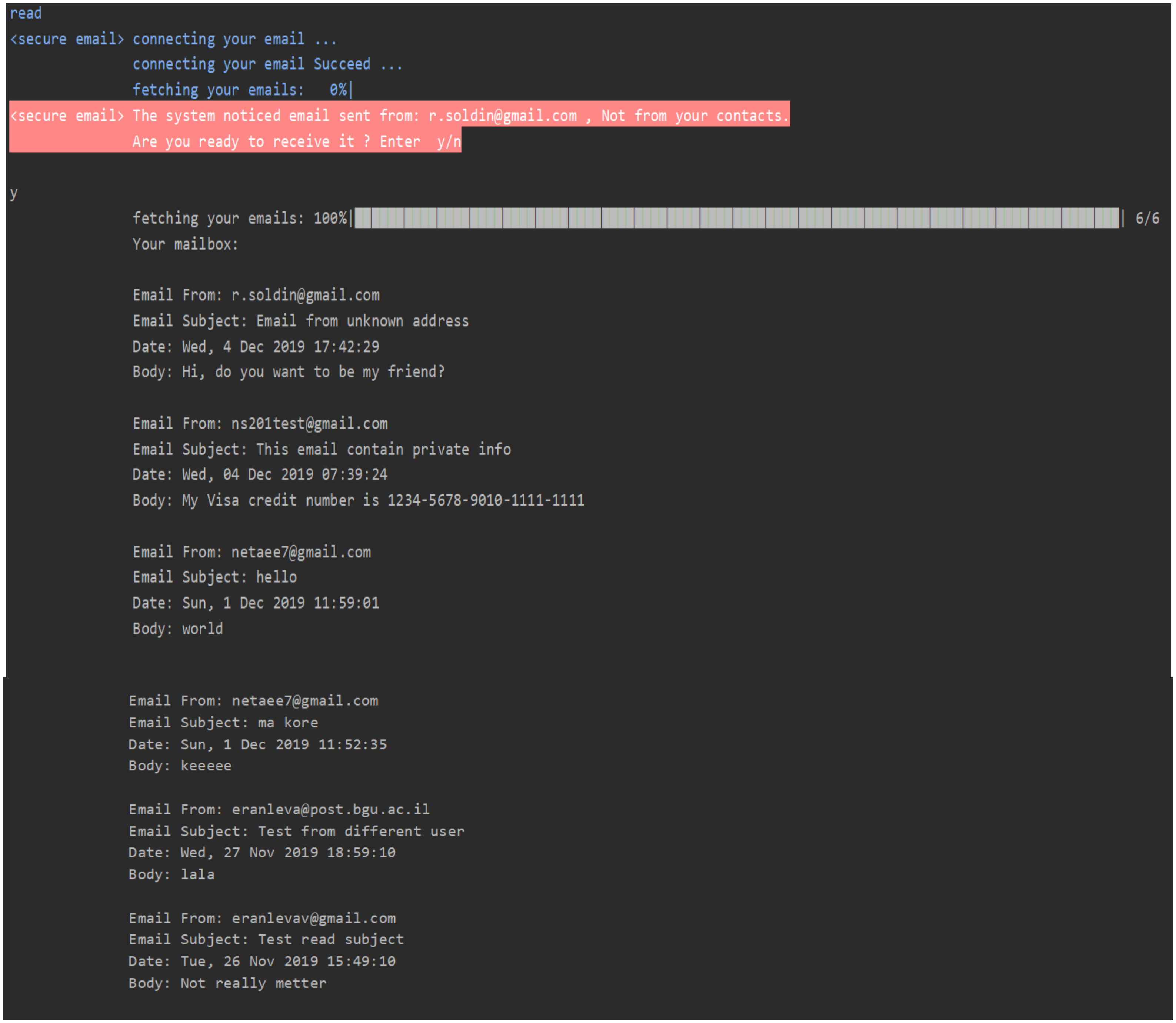
P|#Ylvd#fuhglw#qxpehu#lv#4567089:;0<3430444404444



```
read
<secure email> connecting your email ...
               connecting your email Succeed ...
               fetching your emails: 100%|██████████████████████████████████████████████████| 5/5
               Your mailbox:

               Email From: ns201test@gmail.com
               Email Subject: This email contain private info
               Date: Wed, 04 Dec 2019 07:39:24
               Body: My Visa credit number is 1234-5678-9010-1111-1111
```

In some cases, the user will want to send messages with sensitive information via Email. In this case, if the user account was hacked the hacker isn't expose to the data – as shown in the picture above, the data can be read only through our system.

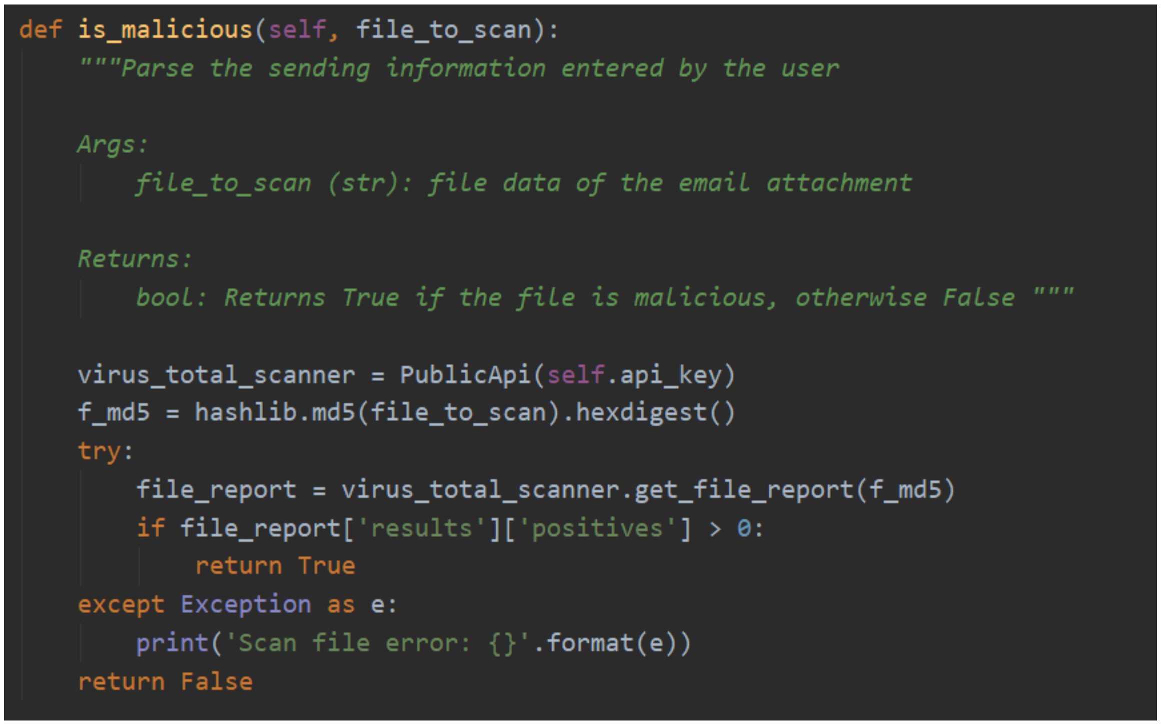
Scenario 5 - Receiving a message



```
read
<secure email> connecting your email ...
               connecting your email Succeed ...
               fetching your emails:   0%|
<secure email> The system noticed email sent from: r.soldin@gmail.com , Not from your contacts.
               Are you ready to receive it ? Enter  y/n

y
               fetching your emails: 100%|██████████████████████████████████████████████| 6/6
               Your mailbox:

               Email From: r.soldin@gmail.com
               Email Subject: Email from unknown address
               Date: Wed, 4 Dec 2019 17:42:29
               Body: Hi, do you want to be my friend?

               Email From: ns201test@gmail.com
               Email Subject: This email contain private info
               Date: Wed, 04 Dec 2019 07:39:24
               Body: My Visa credit number is 1234-5678-9010-1111-1111

               Email From: netaee7@gmail.com
               Email Subject: hello
               Date: Sun, 1 Dec 2019 11:59:01
               Body: world


               Email From: netaee7@gmail.com
               Email Subject: ma kore
               Date: Sun, 1 Dec 2019 11:52:35
               Body: keeeee

               Email From: eranleva@post.bgu.ac.il
               Email Subject: Test from different user
               Date: Wed, 27 Nov 2019 18:59:10
               Body: lala

               Email From: eranlevav@gmail.com
               Email Subject: Test read subject
               Date: Tue, 26 Nov 2019 15:49:10
               Body: Not really metter
```

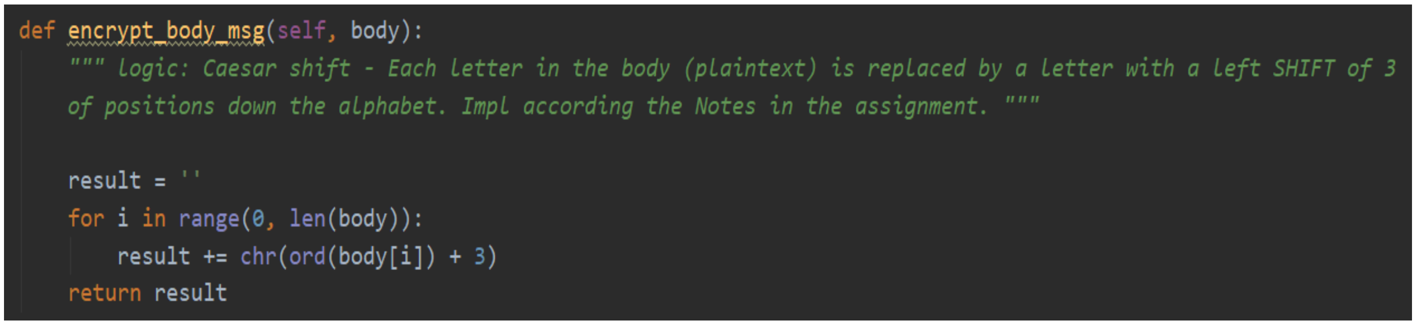
In the picture above we can see a scenario of an unknown address (r.soldin@gmail.com) trying to send a message to our user account. The system asks the user whether to accept the message or to ignore it.

Important methods



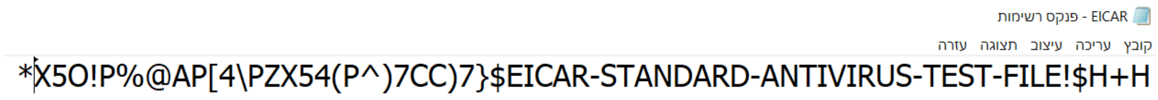
```python
def is_malicious(self, file_to_scan):
    """Parse the sending information entered by the user

    Args:
        file_to_scan (str): file data of the email attachment

    Returns:
        bool: Returns True if the file is malicious, otherwise False """

    virus_total_scanner = PublicApi(self.api_key)
    f_md5 = hashlib.md5(file_to_scan).hexdigest()
    try:
        file_report = virus_total_scanner.get_file_report(f_md5)
        if file_report['results']['positives'] > 0:
            return True
    except Exception as e:
        print('Scan file error: {}'.format(e))
    return False
```

The function above scans the files attached to the message using VirusTotal. We chose VirusTotal because it's scan a file using multi anti virus softwares – for maximum security.



```python
def encrypt_body_msg(self, body):
    """ logic: Caesar shift - Each letter in the body (plaintext) is replaced by a letter with a left SHIFT of 3
    of positions down the alphabet. Impl according the Notes in the assignment. """

    result = ''
    for i in range(0, len(body)):
        result += chr(ord(body[i]) + 3)
    return result
```

This function encrypt a message body using ceasar cipher encryption method. We chose to use ceasar cipher because of it's simplicity and efficiency. In case of future development we can add or change the encryption method for more complex and stronger encryption.



```
*X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H
```

This is the file as a fake malicious file for testing our system.