20BCP055

Python script for email forensics

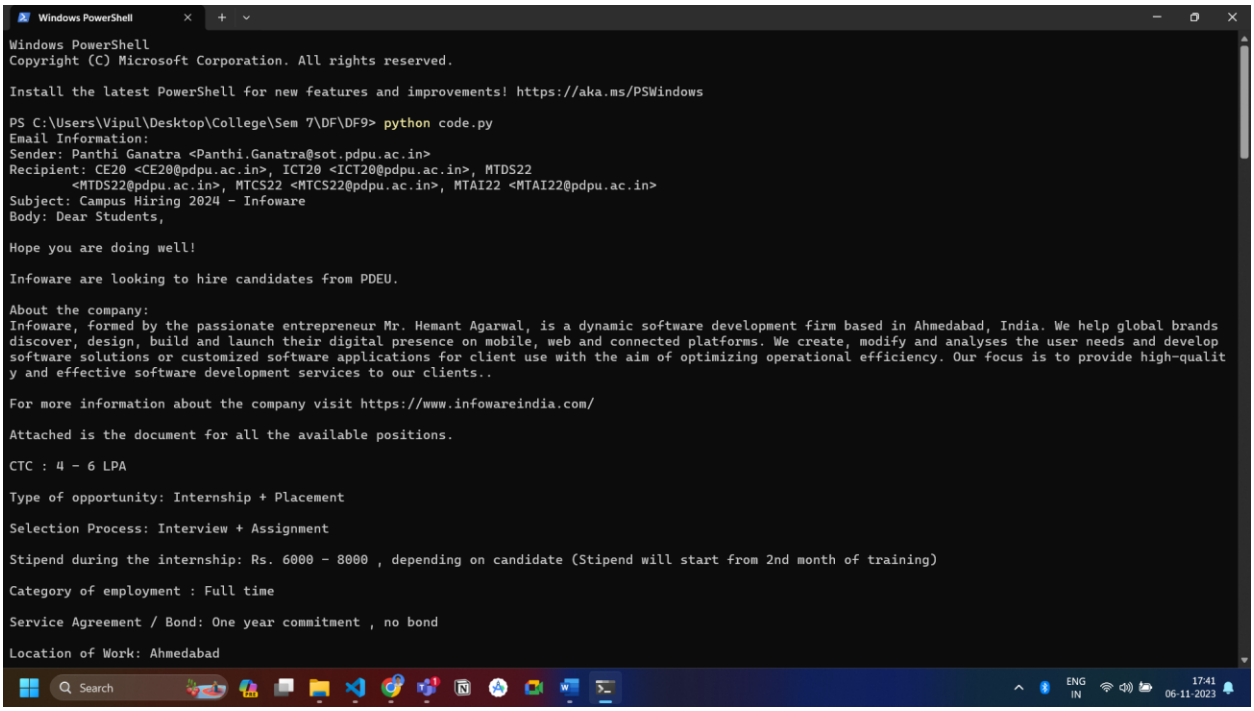
Extraction of Information like,

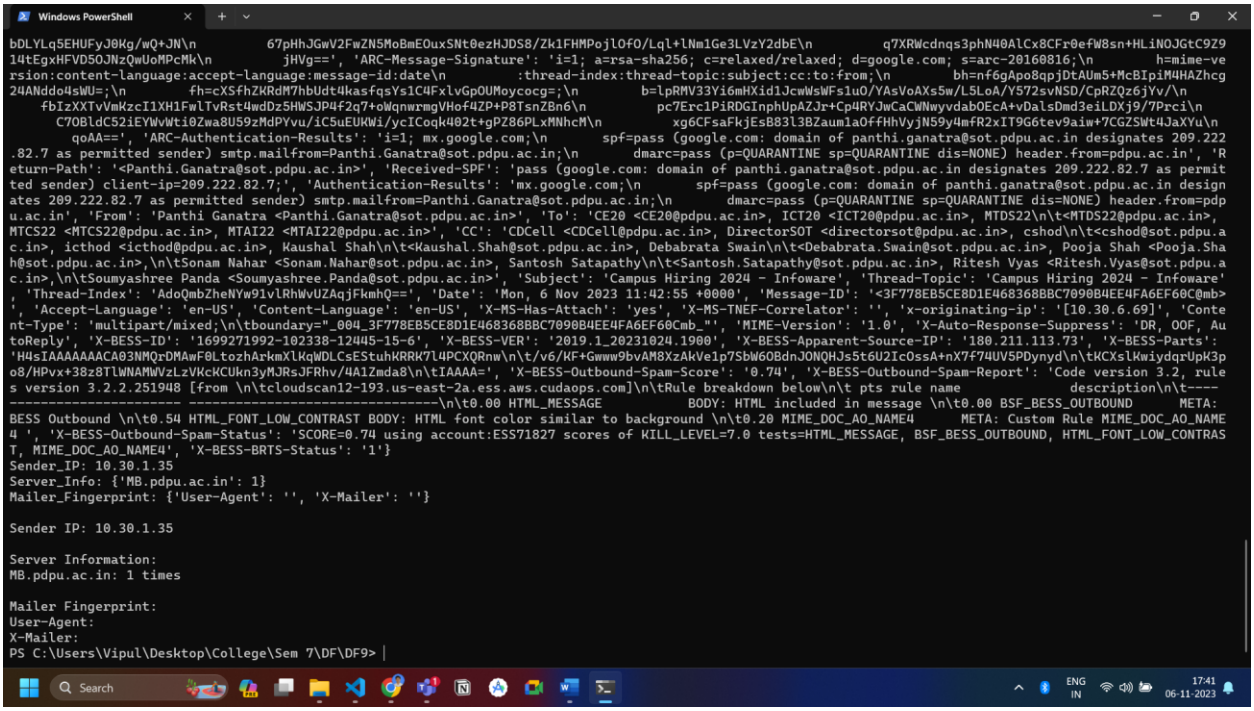
- **Header Analysis**
- **Server investigation**
- **Network Device Investigation**
- **Sender Mailer Fingerprints**

from EML files

```
import email
import os
import re

def extract_email_info(eml_file):
    with open(eml_file, 'r', encoding='utf-8') as eml:
        msg = email.message_from_file(eml)

        sender = msg['From']
        recipient = msg['To']
        subject = msg['Subject']

        body = ""
        if msg.is_multipart():
            for part in msg.walk():
                content_type = part.get_content_type()
                content_disposition = str(part.get("Content-Disposition"))
                try:
                    body += part.get_payload(decode=True).decode()
                except Exception as e:
                    pass
        else:
            body = msg.get_payload(decode=True).decode()

        headers = dict(msg.items())
        sender_ip = extract_sender_ip(headers)
        server_info = get_server_information(headers)
        mailer_fingerprint = extract_mailer_fingerprint(headers)

        return {
            'Sender': sender,
            'Recipient': recipient,
            'Subject': subject,
            'Body': body,
            'Headers': headers,
            'Sender_IP': sender_ip,
            'Server_Info': server_info,
            'Mailer_Fingerprint': mailer_fingerprint
        }

def extract_sender_ip(headers):
    received_headers = headers.get("Received", "")
    ip_pattern = r'\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b'
    match = re.search(ip_pattern, received_headers)
    if match:
        return match.group()
```

```
    return None

def get_server_information(headers):
    server_info = {}
    received_headers = headers.get("Received", "").split('\n')
    for header in received_headers:
        server_match = re.search(r'from\s+(\S+)', header)
        if server_match:
            server = server_match.group(1)
            if server not in server_info:
                server_info[server] = 1
            else:
                server_info[server] += 1
    return server_info

def extract_mailer_fingerprint(headers):
    user_agent = headers.get("User-Agent", "")
    x_mailer = headers.get("X-Mailer", "")
    return {
        "User-Agent": user_agent,
        "X-Mailer": x_mailer
    }

if __name__ == "__main__":
    eml_file_path = 'path/to/your/eml/file.eml'

    if os.path.exists(eml_file_path):
        email_info = extract_email_info(eml_file_path)
        print("Email Information:")
        for key, value in email_info.items():
            print(f'{key}: {value}')

        print("\nSender IP:", email_info['Sender_IP'])

        print("\nServer Information:")
        for server, count in email_info['Server_Info'].items():
            print(f'{server}: {count} times')

        print("\nMailer Fingerprint:")
        for key, value in email_info['Mailer_Fingerprint'].items():
            print(f'{key}: {value}')
    else:
        print(f"File not found: {eml_file_path}")
```





Structuring MBOX files from Google Takeout using Python

```
import mailbox
import os
import email.utils

# Directory containing MBOX files from Google Takeout
mbox_dir = 'path/to/your/mbox/directory'
```

```
# Directory to store structured emails
output_dir = 'path/to/output/directory'

def save_email_as_file(email_message, output_dir):
    # Create a unique filename based on the email's date and subject
    date_tuple = email.utils.parsedate(email_message['Date'])
    if date_tuple:
        email_date = email.utils.formatdate(email.utils.mktime_tz(date_tuple))
    else:
        email_date = 'unknown_date'

    subject = email_message['Subject'] or 'unknown_subject'

    email_filename = os.path.join(output_dir, f"{email_date}_{subject}.eml")

    # Save the email as a .eml file
    with open(email_filename, 'wb') as email_file:
        email_file.write(email_message.as_bytes())

def structure_mbox(mbox_dir, output_dir):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    mbox = mailbox.mbox(mbox_dir)

    for message in mbox:
        save_email_as_file(message, output_dir)

if __name__ == "__main__":
    structure_mbox(mbox_dir, output_dir)
```