

Peer Review Report for Group 30 – Jakob Sellmann

Dynamic Analysis

Functional Tests:

Dynamic analysis involved running the provided main.py and onp_node.py, Both python code were successfully transmitted chat messages between different node.

```
Enter a command: message
Usage: message <recipient> <message_content>
Enter a command: discover 127.0.0.1:8888
[DEBUG] Connection from ('127.0.0.1', 59401) received
Enter a command: [DEBUG] Received message: {'type': 'discovery', 'node': ['localhost', 8888]}
Discovered new neighbor: ('localhost', 8888)
[DEBUG] Routing table updated: {'localhost', 8888}: ('localhost', 8888)
message 127.0.0.1:8888 "Hello, Node 8888!"
[DEBUG] Attempting to forward message to ('127.0.0.1', 8888) with content: {'recipient': '127.0.0.1:8888', 'content': '9zx5IKHwMrd
ng2NULAbvex39tXOnAaWggrpCijcwUgPTE=', 'type': 'message'}
[DEBUG] Message successfully forwarded to ('127.0.0.1', 8888)
[DEBUG] Connection from ('127.0.0.1', 59417) received
Enter a command: [DEBUG] Received message: {'recipient': '127.0.0.1:8888', 'content': '9zx5IKHwMrdng2NULAbvex39tXOnAaWggrpCijcwU
gPTE=', 'type': 'message'}
Handling message for recipient: 127.0.0.1:8888
Current node: localhost:8888 (Normalized: 127.0.0.1:8888)
Decrypted message: "Hello, Node 8888!"

Enter a command: message 127.0.0.1:8888 "Hello, Node 8888!"
[DEBUG] Attempting to forward message to ('127.0.0.1', 8888) with content: {'recipient': '127.0.0.1:8888', 'content': 'CR8PkOfHNaBlTNeABTeqSpHqQaOLUG2nEyiURZenMooErE=', 'type': 'message'}
[DEBUG] Message successfully forwarded to ('127.0.0.1', 8888)
[DEBUG] Connection from ('127.0.0.1', 59421) received
Enter a command: [DEBUG] Received message: {'recipient': '127.0.0.1:8888', 'content': 'CR8PkOfHNaBlTNeABTeqSpHqQaOLUG2nEyiURZenMooErE=', 'type': 'message'}
Handling message for recipient: 127.0.0.1:8888
Current node: localhost:8888 (Normalized: 127.0.0.1:8888)
Decrypted message: "Hello, Node 8888!"
```

Vulnerabilities

- **Backdoor 1 - Weak Encryption Key:** The VulnerableONPNode uses a hardcoded 16-byte AES key (b'1234567890123456'). Hardcoding keys is a severe security vulnerability, as anyone with access to the code can decrypt all messages.

```
class VulnerableONPNode(SecureONPNode):
    def __init__(self, host, port):
        # Hardcoded weak encryption key (intentionally vulnerable)
        weak_key = b'1234567890123456'
        super().__init__(host, port, weak_key)
```

- **Backdoor 2 - Hardcoded Backdoor Login:** The login function provides access via a hardcoded username and password (admin and backdoor123). This backdoor enables attackers to bypass conventional authentication and obtain access to the system without knowing their real credentials.

```
# Hardcoded backdoor login
def login(self, username, password):
    if username == "admin" and password == "backdoor123":
        print("Backdoor access granted!")
        return True
    return super().login(username, password)
```

Strengths:

- The node discovers neighbors and handles communication effectively.
- AES encryption (EAX mode) is a good method for securing communication. The messages are encrypted and base64 encoded.

Weaknesses:

- The lack of input validation and authentication in main.py and onp_node.py exposes the system to a number of vulnerabilities.
- The possibility of a hardcoded weak key and backdoor in vulnerabilities.py is a critical vulnerability that would let an attacker to overcome encryption and get unauthorized access.

Recommendations

- Introduce an authentication mechanism for nodes before allowing them to join the network. Implement checks to ensure the authenticity of neighbor nodes.
- Remove the hardcoded backdoor and weak encryption key.

Thanks for the work hard on the chat application, the function is work pretty good.