Dynamic Analysis Report

1. Analysis Environment

This dynamic analysis was conducted in a Docker container with network access disabled (--network=none). Tools such as strace, inotify, and ss were used to monitor the program's behavior for 60 seconds. The target system is a Python-based chat server (server.py), running in a Python 3.11 environment.

2. Runtime Behavior

1. Startup Phase

The program successfully launched and printed a warning indicating that websockets.server.WebSocketServerProtocol is deprecated. This shows that the server initialized correctly and entered a listening state.

2. Network Listening

According to ss.txt output, the application listens on 127.0.0.1:8080 and [::1]:8080. It binds only to the localhost interface and does not expose any external network ports.

3. System Call Behavior

The strace logs indicate the program executed typical server-related system calls such as socket(), bind(), listen(), futex(), and epoll_wait(). No suspicious connect() attempts or access to sensitive system files were observed.

3. Security Assessment

Check Item	Result	Explanation
Network Exposure	Safe	Listens only on 127.0.0.1:8080, not externally accessible.
External Communication	Safe	No connect() or outbound network activity detected.
File Access	Safe	Access limited to project files, no sensitive paths accessed.
Command Execution	Safe	No execve or shell command executions found.
Cryptographic Modules	Caution	Uses cryptography and argon2; ensure secure key handling.

Library Deprecation	Info	websockets library uses a deprecated class; upgrade recommended.

4. Improvement Suggestions

1. Upgrade the websockets library to replace the deprecated WebSocketServerProtocol class.
2. Implement structured logging to improve monitoring and traceability.
3. Optionally test under a non-isolated network environment to verify whether the program attempts to connect to nodes listed in bootstrap.yaml.
4. Maintain least-privilege and read-only file system configurations to further enhance security.

5. Conclusion

The dynamic analysis results indicate that this Python chat system is secure overall. The program listens only on the localhost interface and does not attempt external communication or execute shell commands. System calls and file access patterns are normal, and no suspicious activity was observed. Overall security risk is low.