Dynamic Analysis Report - Impl15 (SOCP Flask Chat Server)

1. Execution Context

Container: run-impl15
Environment: Non-root Docker sandbox, network disabled (--network=none), Python 3.11 on Debian bookworm slim.
Execution time: 60 seconds (timeout 60s)
Main entry: run.py (Flask-based chat and WebSocket service).
Goal: Execute Flask chat application in isolation to capture runtime behavior, file access, and system calls.

2. File-System Activity (from program.log and strace)

The Flask chat server executed in the sandbox environment, importing standard modules and initializing application components. Normal Python file access was observed via openat(), read(), and write() calls for module imports and local caching.

No evidence of file creation or modification outside /home/auditor/app. Temporary cache files and Flask runtime logs were created, consistent with standard WSGI behavior.
Verdict: File activity remained within expected sandbox scope.

3. Network Activity (from ss.txt)

The application bound locally to 127.0.0.1:5000 and ws://127.0.0.1:5000/ws for HTTP and WebSocket endpoints. No external network connections were initiated. The ss.txt output confirmed only loopback bindings and no remote peers.
Verdict: No external communication or data leakage occurred.

4. System-Call Trace (from strace_log.17)

System call tracing revealed frequent openat(), read(), and write() operations related to Flask framework initialization, template loading, and local log writes. No connect() to external IPs, no fork() or execve() calls for subprocess creation were detected. The program executed under stable conditions and terminated cleanly after the timeout period.
Verdict: Stable I/O and network isolation verified.

5. Security Observations

Category	Observation	Severity
Execution	Flask server runs locally, graceful shutdown.	None
File I/O	Limited to internal directories and logs.	None

Network	Loopback binding only; no outbound connections.	None
Process	Single PID, no privilege escalation or shell spawning.	None

6. Overall Assessment

Behavior: Benign — isolated Flask server performing as designed.

Security posture: Strong — sandbox containment verified, no data leakage or remote access.

Reliability: Stable — consistent process behavior and proper resource cleanup.

7. Recommendations

1. Continue testing under --network=none for future code submissions to validate isolation.

2. If remote networking is ever enabled, restrict the bind interface to 127.0.0.1 explicitly.

3. Implement authentication and rate limiting for the WebSocket endpoint in production builds.

4. Introduce log rotation to avoid unbounded file growth over long sessions.

5. Periodically review Flask dependencies for security patches.