Interview Exercise

This exercise is designed to assess your problem-solving skills, coding style, and basic familiarity with Python. Please follow the instructions carefully and submit your solution within the specified time.

Overview

You are tasked with building a minimalistic **Command and Control (C2)** simulation, consisting of two separate Python programs:

- A **server** (command dispatcher)
- A **client** (agent)

The design should be modular, readable, and well-structured, with clear separation of responsibilities between the server and the client.

What Is a C2 System?

A C2 server (Command and Control server) is a central component used by red teams to manage and control compromised systems during an operation. It acts as the “command center,” allowing operators to issue commands, retrieve data, and control malicious tools on target systems remotely. The server communicates with implanted agents (beacons) in the target network, facilitating tasks like data exfiltration, lateral movement, and persistence.

For this assignment, you will build a minimal C2 system. It will use basic TCP communication and shell command execution.

Level 1 -Basic Server & Client

- Server (Python): listen on a TCP port and accept client connections.
- Client (Python): connect to server and send a registration message (ID or hostname).
- On registration the server opens a session so the operator can send system commands to that client.
- Client executes received commands locally and sends back results.
- Server displays received results to the user (include both stdout and stderr).
- Operator can send another command and repeat.

Level 2 - Add logging

- Log the following (server side):
  - Client connection events
  - Registration messages
  - Commands sent to clients
  - Responses received from clients
- Log both to the console and to a file with timestamps and session IDs.

Level 3 - Multi-client support & session switching

- Make the server support concurrent connections (threading or async).
- Maintain separate sessions per client and allow the operator to list/switch between sessions to send commands to a specific client.
- Ensure logs remain correct per session.

Level 4 - Bonus: Encryption & Authentication

- Add authentication so clients and server validate each other (e.g., token or HMAC).
- Add encryption to secure communication (e.g., AES or TLS).

Deliverables

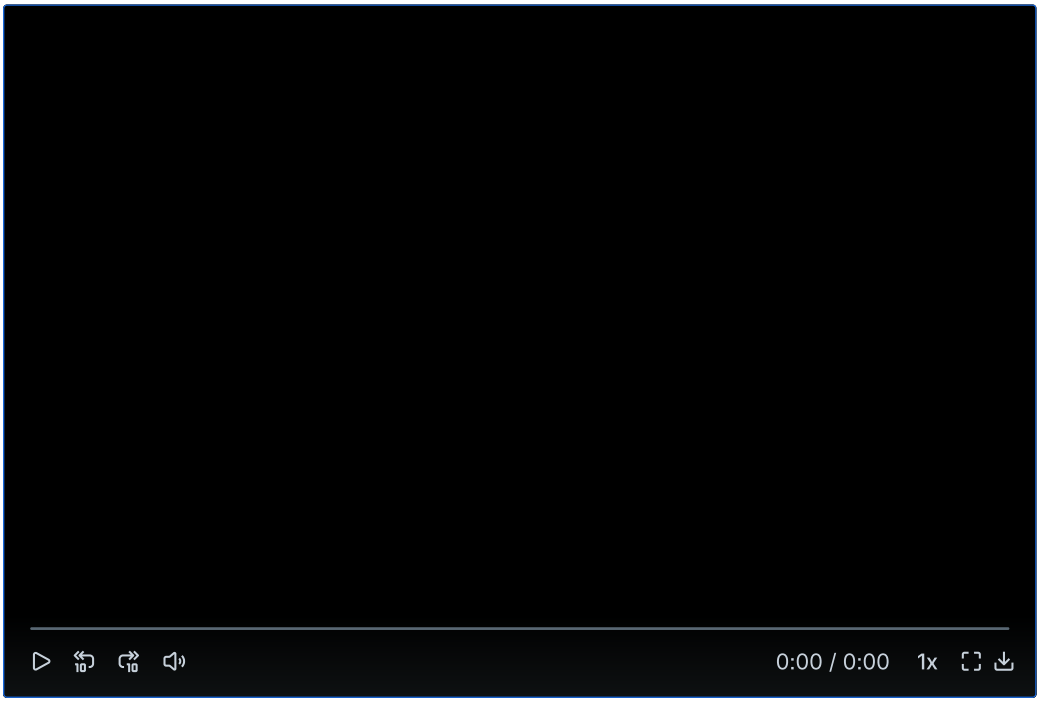
Please submit the following:

- GitHub Repository containing:
  - A concise README file with the following sections:
    - Setup Instructions
    - Instructions for Running the Server and Client
    - Description of the Protocol
    - Overview of any Optional Features Implemented

Guidelines

- Use **Python 3.7+**
- Follow good coding practices (naming, structure, comments)
- Keep it simple, readable, and modular

Note: Don’t worry about completing all levels. We don’t expect you to finish everything - just do as much as you can. Submitting partial work is absolutely fine!



demo