Agentic Honeypot System Overview

This document explains the current honeypot implementation in simple words. It describes what the system does, how requests flow through the code, and when the GUVI callback is triggered.

Purpose

The service accepts incoming messages, detects scam intent, engages the scammer with a believable reply, extracts scam intelligence, and finally reports the results to the GUVI evaluation endpoint.

Key Entry Points

- HTTP server: index.js (Express app)
- API route: POST /api/v1/message (routes/message.route.js)
- Auth check: authMiddleware (middleware/authMiddleware.js)
- Controller: handleMessage (controllers/message.controller.js)
- Core logic: processMessage (services/honeypot.service.js)

Main Components

- Session store: in-memory Map with per-session state (storage/conversationStore.js)
- LLM client: OpenAI Responses API wrapper (utils/llmClient.js)
- Prompts: classifier, agent reply, extraction, end decision (utils/prompts.js)
- Callback: GUVI final result POST with retries (services/callback.service.js)

Session State (Per sessionId)

- messages[]: full conversation (scammer + honeypot)
- scamAssessment: result of the first classification
- extractedIntelligence: bankAccounts, upiIds, links, phones, keywords, case/staff IDs, names
- scamSignals: claimed organization/department, scam type, tactics
- dialogState: what has been asked already and which targets are missing
- counters: lastExtractedMessageCount, extractionRuns, total messages
- flags: endConversation, callbackSent

How a Message Is Processed

1. Request arrives at POST /api/v1/message.
2. authMiddleware validates x-api-key. If missing or wrong, returns 400 with a failed response.
3. handleMessage passes the JSON body to processMessage.
4. processMessage validates sessionId and message.text. If invalid, returns 400 failed response.
5. Session is loaded or created (getOrCreateSession).
6. Incoming message is normalized and appended to messages[].
7. If scamAssessment is not set, the classifier prompt is called once and stored.
8. If scamLikely is false, the service returns a safe response (Message is not likely a scam).
9. If scamLikely is true: the agent reply prompt is called. The reply is appended and dialogState is updated.
10. Every 3 new messages (or at end), the extraction prompt is called to update intelligence.
11. End decision logic checks thresholds and may call the end-decision prompt.
12. If endConversation is true, a final extraction is run if needed, then a disengagement reply is returned and the GUVI callback is sent.

End Conditions (Simplified)

- Minimum total messages, minimum scammer messages, and minimum extraction runs must be met.
- Early stop can trigger when all primary intel is collected.
- Hard stop triggers after MIN_TOTAL_MESSAGES + GRACE_MESSAGES.

GUVI Callback

When the conversation ends and scam intent is confirmed, the service posts the final payload to:

https://hackathon.guvi.in/api/updateHoneyPotFinalResult

- Payload includes: sessionId, scamDetected, totalMessagesExchanged, extractedIntelligence, agentNotes.
- Callback is retried up to 3 times on failure with a short backoff.
- After callback, callbackSent is set to true and the session hard-stops future replies.

Simulated Workflow Example

This is a simulated flow showing which functions are called and when, from the first message to the GUVI callback.

Turn 1: First scam message

1. POST /api/v1/message
2. authMiddleware checks x-api-key
3. handleMessage -> processMessage
4. getOrCreateSession(sessionId)
5. appendMessage(scammer message)
6. buildClassifierPrompt -> generateJson (LLM) -> scamAssessment stored
7. buildAgentReplyPrompt -> generateJson (LLM) -> honeypot reply
8. appendReply + updateDialogState
9. return { status: 'success', reply: '...' }

Turn 2-3: Continued scam messages

1. appendMessage (new scammer message)
2. buildAgentReplyPrompt -> generateJson -> reply
3. appendReply + updateDialogState
4. Every 3 messages: buildIntelligenceExtractionPrompt -> generateJson -> updateIntelligence + markExtractionRun

Turn N: End decision

1. Check thresholds: total messages, scammer messages, extraction runs
2. If end gate hit: buildConversationEndPrompt -> generateJson
3. If endConversation=true: final extraction if needed
4. Append disengagement reply: 'Alright, I have what I need for now. I'll follow up shortly.'
5. sendFinalResult(payload) -> GUVI endpoint
6. Set callbackSent=true and endConversation=true

Notes About Timing Logs

Response time is measured from request start to response creation. It includes LLM calls but not the external scammer delays.

Configuration (Env Vars)

- API_KEY: required for x-api-key validation
- OPENAI_API_KEY: LLM access
- OPENAI_MODEL: model name for Responses API
- MIN_TOTAL_MESSAGES, MIN_SCAMMER_MESSAGES, MIN_EXTRACTION_RUNS, GRACE_MESSAGES