

# Agentic Honey-Pot for Scam Detection & Intelligence Extraction

## 1. Introduction

Online scams such as bank fraud, UPI fraud, phishing, and fake offers are becoming increasingly adaptive. Scammers change their tactics based on user responses, making traditional detection systems ineffective.

This challenge requires participants to build an Agentic Honey-Pot — an AI-powered system that detects scam intent and autonomously engages scammers to extract useful intelligence without revealing detection.

## 2. Objective

Design and deploy an AI-driven honeypot system that can:

- Detect scam or fraudulent messages
- Activate an autonomous AI Agent
- Maintain a believable human-like persona
- Handle multi-turn conversations
- Extract scam-related intelligence
- Return structured results via an API

## 3. What You Need to Build

Participants must deploy a public REST API that:

- Accepts incoming message events
- Detects scam intent
- Hands control to an AI Agent
- Engages scammers autonomously
- Extracts actionable intelligence
- Returns a structured JSON response

- Secures access using an API key

## 4. API Authentication

- x-api-key: YOUR\_SECRET\_API\_KEY
- Content-Type: application/json

## 5. Evaluation Flow

1. Platform sends a suspected scam message
2. Your system analyzes the message
3. If scam intent is detected, the AI Agent is activated
4. The Agent continues the conversation
5. Intelligence is extracted and returned
6. Performance is evaluated

## 6. API Request Format (Input)

Each API request represents one incoming message in a conversation.

### 6.1 First Message (Start of Conversation)

This is the initial message sent by a suspected scammer. There is no prior conversation history.

```
{  
  "sessionId": "wertyu-dfghj-ertyui",  
  "message": {  
    "sender": "scammer",  
    "text": "Your bank account will be blocked today. Verify immediately.",  
    "timestamp": 1770005528731  
  },  
  "conversationHistory": [],
```

```
"metadata": {  
  "channel": "SMS",  
  "language": "English",  
  "locale": "IN"  
}  
}
```

## 6.2 Second Message (Follow-Up Message)

This request represents a continuation of the same conversation. Previous messages are now included in `conversationHistory`.

```
{  
  "sessionId": "wertyu-dfghj-ertyui",  
  "message": {  
    "sender": "scammer",  
    "text": "Share your UPI ID to avoid account suspension.",  
    "timestamp": 1770005528731  
  },  
  "conversationHistory": [  
    {  
      "sender": "scammer",  
      "text": "Your bank account will be blocked today. Verify immediately.",  
      "timestamp": 1770005528731  
    },  
    {
```

```
    "sender": "user",

    "text": "Why will my account be blocked?",

    "timestamp": 1770005528731

  }

],

"metadata": {

  "channel": "SMS",

  "language": "English",

  "locale": "IN"

}

}
```

### 6.3 Request Body Field Explanation

#### message (Required)

The latest incoming message in the conversation.

Field	Description
sender	scammer or user
text	Message content
timestamp	ISO-8601 format

#### conversationHistory (Optional)

All previous messages in the same conversation.

- Empty array ( `[]` ) for first message
- Required for follow-up messages

**metadata** (Optional but Recommended)

Field	Description
channel	SMS / WhatsApp / Email / Chat
language	Language used
locale	Country or region

## 7. Agent Behavior Expectations

The AI Agent must:

- Handle multi-turn conversations
- Adapt responses dynamically
- Avoid revealing scam detection
- Behave like a real human
- Perform self-correction if needed





## 8. Agent output should be like

```
{  
  
  "status": "success",  
  
  "reply": "Why is my account being suspended?"  
}
```

## 9. Evaluation Criteria

- Scam detection accuracy
- Quality of agentic engagement
- Intelligence extraction
- API stability and response time
- Ethical behavior

## 10. Constraints & Ethics

-  No impersonation of real individuals
-  No illegal instructions
-  No harassment
-  Responsible data handling

## 11. One-Line Summary

Build an AI-powered agentic honeypot API that detects scam messages, handles multi-turn conversations, and extracts scam intelligence without exposing detection.

## 12. Mandatory Final Result Callback (Very Important)

Once the system **detects scam intent** and the **AI Agent completes the engagement**, participants must **send the final extracted intelligence** to the GUVI evaluation endpoint.

This is **mandatory** for evaluation.

### Callback Endpoint

POST <https://hackathon.guvi.in/api/updateHoneyPotFinalResult>

Content-Type: application/json

### Payload to Send

Participants must send the following JSON payload to the above endpoint:

```
{  
  
  "sessionId": "abc123-session-id",
```

```
"scamDetected": true,

"totalMessagesExchanged": 18,

"extractedIntelligence": {

  "bankAccounts": ["XXXX-XXXX-XXXX"],

  "upIds": ["scammer@upi"],

  "phishingLinks": ["http://malicious-link.example"],

  "phoneNumbers": ["+91XXXXXXXXXX"],

  "suspiciousKeywords": ["urgent", "verify now", "account blocked"]

},

"agentNotes": "Scammer used urgency tactics and payment redirection"

}
```

## When Should This Be Sent?

You must send this **only after**:

1. Scam intent is confirmed (`scamDetected = true`)
2. The AI Agent has completed sufficient engagement
3. Intelligence extraction is finished

This should be treated as the **final step** of the conversation lifecycle.

---

## Field Explanation

Field	Description
sessionId	Unique session ID received from the platform for this conversation

scamDetected	Whether scam intent was confirmed
totalMessagesExchanged	Total number of messages exchanged in the session
extractedIntelligence	All intelligence gathered by the agent
agentNotes	Summary of scammer behavior

### Important Rules

- This callback is **mandatory for scoring**
- If this API call is not made, the solution **cannot be evaluated**
- The platform uses this data to measure:
  - Engagement depth
  - Intelligence quality
  - Agent effectiveness

### Example Implementation (Python)

```
intelligence_dict = {  
  
    "bankAccounts": intelligence.bankAccounts,  
  
    "upilds": intelligence.upilds,  
  
    "phishingLinks": intelligence.phishingLinks,  
  
    "phoneNumbers": intelligence.phoneNumbers,  
  
    "suspiciousKeywords": intelligence.suspiciousKeywords  
  
}  
  
payload = {
```



```
"sessionId": session_id,  
  
"scamDetected": scam_detected,  
  
"totalMessagesExchanged": total_messages,  
  
"extractedIntelligence": intelligence_dict,  
  
"agentNotes": agent_notes  
}  
  
response = requests.post(  
  
    "https://hackathon.guvi.in/api/updateHoneyPotFinalResult",  
  
    json=payload,  
  
    timeout=5  
  
)
```



### Updated One-Line Summary

Build an AI-powered agentic honeypot API that detects scam messages, engages scammers in multi-turn conversations, extracts intelligence, **and reports the final result back to the GUVI evaluation endpoint.**