NDSS SYMPOSIUM/2026

Artifact Evaluated NDSS Available

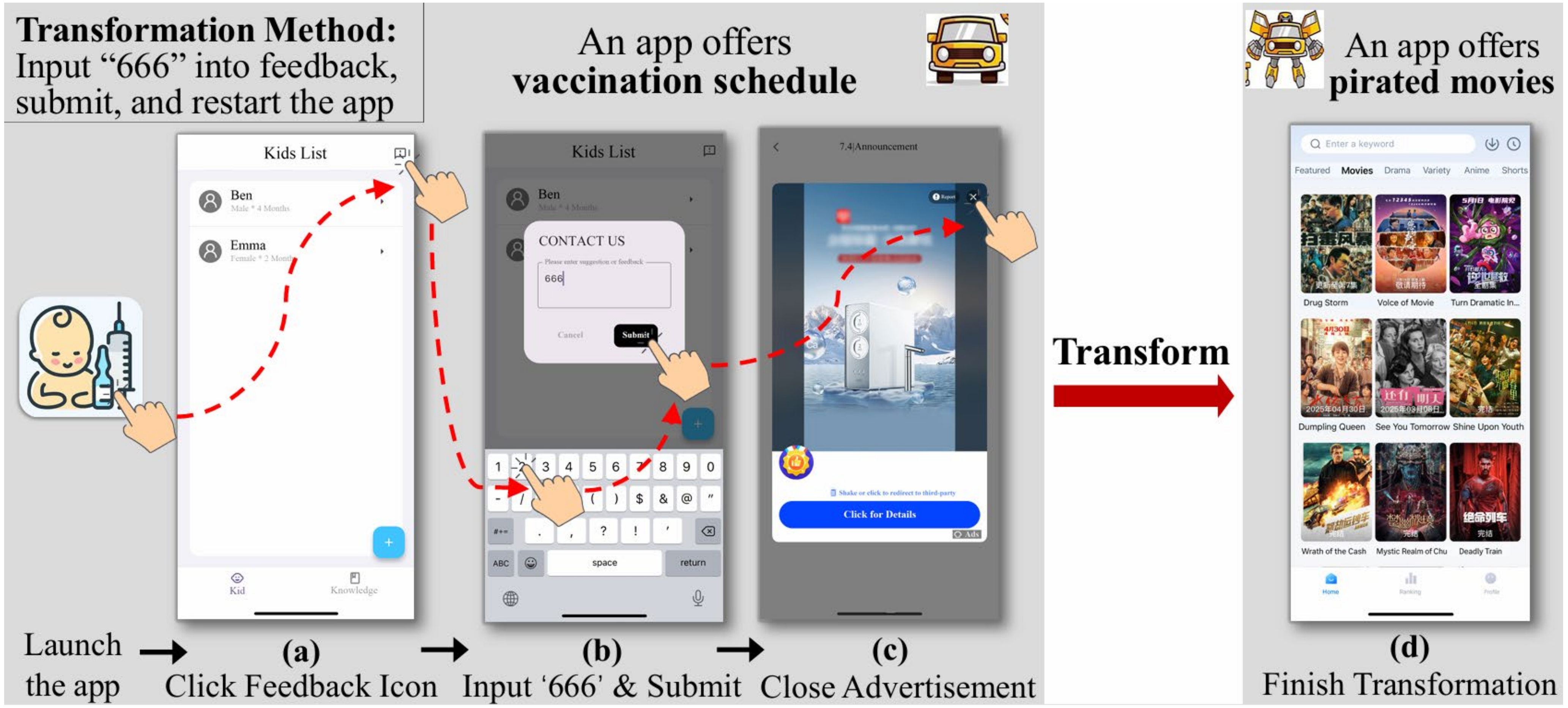
ChameleoScan: Demystifying and Detecting iOS Chameleon Apps via LLM-Powered UI Exploration

Authors: Hongyu Lin*, **Yicheng Hu***, Haitao Xu✉, Yanchen Lu, Mengxia Ren, Shuai Hao, Chuan Yue , Zhao Li, Fan Zhang, Yixin Jiang

* Contribute equally
✉ Corresponding author

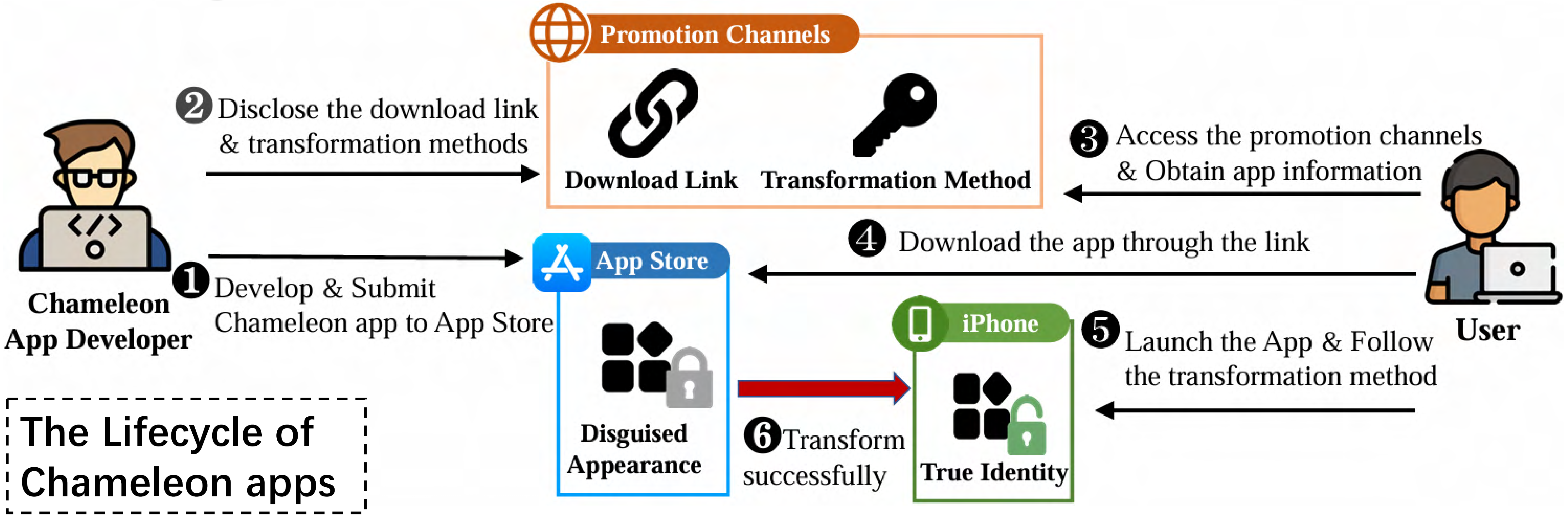
Background — What are Chameleon Apps?

Chameleon apps are apps that **appear legitimate in app stores**, but later **reveal hidden illicit functionality** after user installation.



A Chameleon app's transformation process.

Background — Lifecycle of Chameleon Apps


Promotion Channels
❷ Disclose the download link & transformation methods
Download Link
Transformation Method
❸ Access the promotion channels & Obtain app information
Chameleon App Developer
❶ Develop & Submit Chameleon app to App Store
App Store
❹ Download the app through the link
User
iPhone
❺ Launch the App & Follow the transformation method
Disguised Appearance
❻ Transform successfully
True Identity
The Lifecycle of Chameleon apps

- **Chameleon apps**, distinguished by their **highly covert distribution channels** and **seemingly legitimate storefronts**, are **difficult to detect**.
- Apple removed **38,315 fraudulent** apps in 2024 year alone, as reported in the *App Store Transparency Report.*

Related Work — Existing Detection Approaches

TDSC 2019: "Understanding Illicit UI in iOS apps Through Hidden UI Analysis"

Chameleon-Hunter

- Analyze UI layout files and code control flow to infer **UI transformations**.
- Build a **labeled view controller graph (LVCG)** to identify **hidden UIs**.
- Use **word embeddings / TF-IDF** to validate **UI semantic consistency**.

Static code analysis

Majority

Fails on apps built with **dynamic UI** or **hybrid frameworks.**

Dynamic behavior analysis ✓

Lagging

Fails on apps with **limited metadata** or **novel code patterns**.

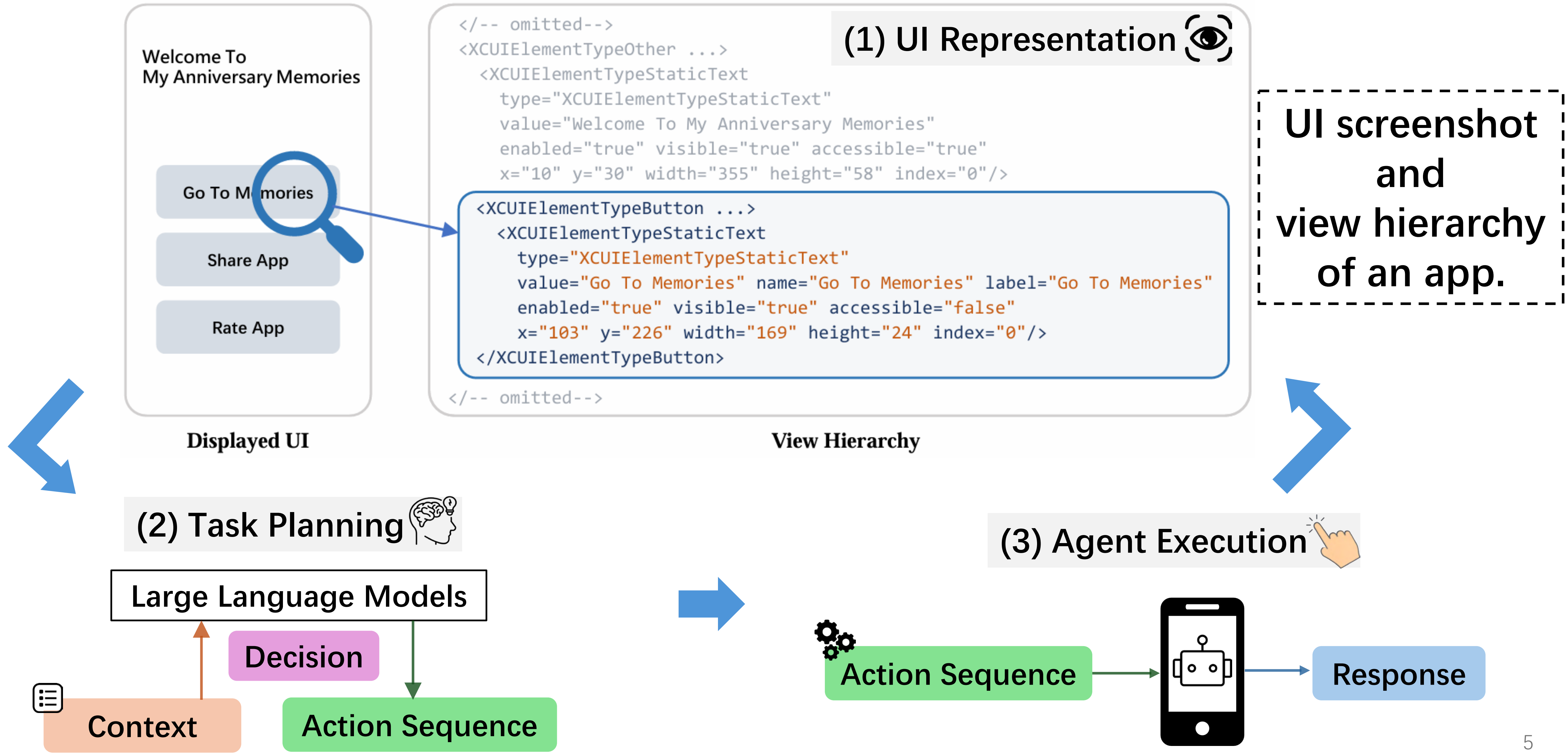
ICPC 2024: "No Source Code? No Problem! Demystifying and Detecting Mask Apps in iOS"

Mask-Catcher

- Analyze **user reviews** to extract semantic labels of the *true* functionality.
- Analyze **recommendation/association links** to surface related suspicious apps.
- Use **code similarity** to confirm membership in a known family.

App metadata analysis

A Straightforward Approach — Mobile Task Automation



```
</-- omitted-->
<XCUIElementTypeOther ...>
  <XCUIElementTypeStaticText
    type="XCUIElementTypeStaticText"
    value="Welcome To My Anniversary Memories"
    enabled="true" visible="true" accessible="true"
    x="10" y="30" width="355" height="58" index="0"/>
  <XCUIElementTypeButton ...>
    <XCUIElementTypeStaticText
      type="XCUIElementTypeStaticText"
      value="Go To Memories" name="Go To Memories" label="Go To Memories"
      enabled="true" visible="true" accessible="false"
      x="103" y="226" width="169" height="24" index="0"/>
  </XCUIElementTypeButton>
</-- omitted-->
```

Displayed UI

View Hierarchy

Preliminary Study — Data Collection

App Metadata

- **App Name:** TranquilAngler
- **App Identifier:** id6479501175 / com.BuiVanBientieu.TranquilAngler
- **App Icon:**
- **App Category:** Health & Fitness
- **App Description:** Tranquil Angler is a fishing app that helps relieve stress and relieve stress……
- **User Review:** "This doesn't work anymore. One star as a warning—fake content. Clicking just shows ads promoting other apps; I can't get in no matter what. Lollllll."

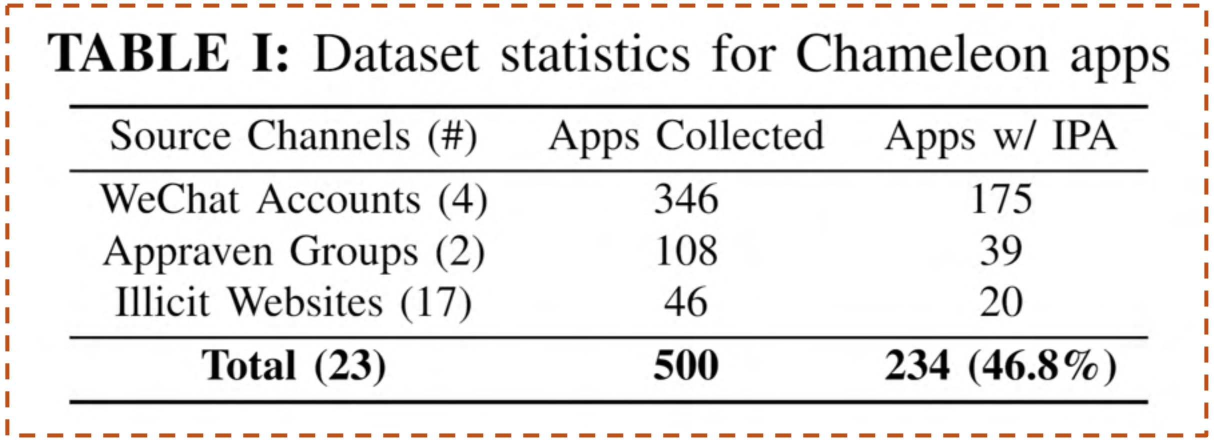


TABLE I: Dataset statistics for Chameleon apps

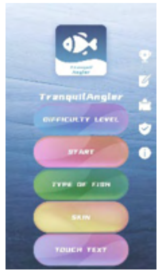
Source Channels (#)	Apps Collected	Apps w/ IPA
WeChat Accounts (4)	346	175
Appraven Groups (2)	108	39
Illicit Websites (17)	46	20
Total (23)	**500**	**234 (46.8%)**

The app itself

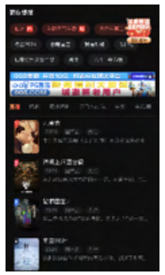
- **IPA Package**

Transformation Method (example)

Disguised Functionality
Displayed Most of the Time



App Launched at Midnight
<Trigger Condition>



Hidden Functionality
Revealed Only at Midnight

Preliminary Study — Characterization

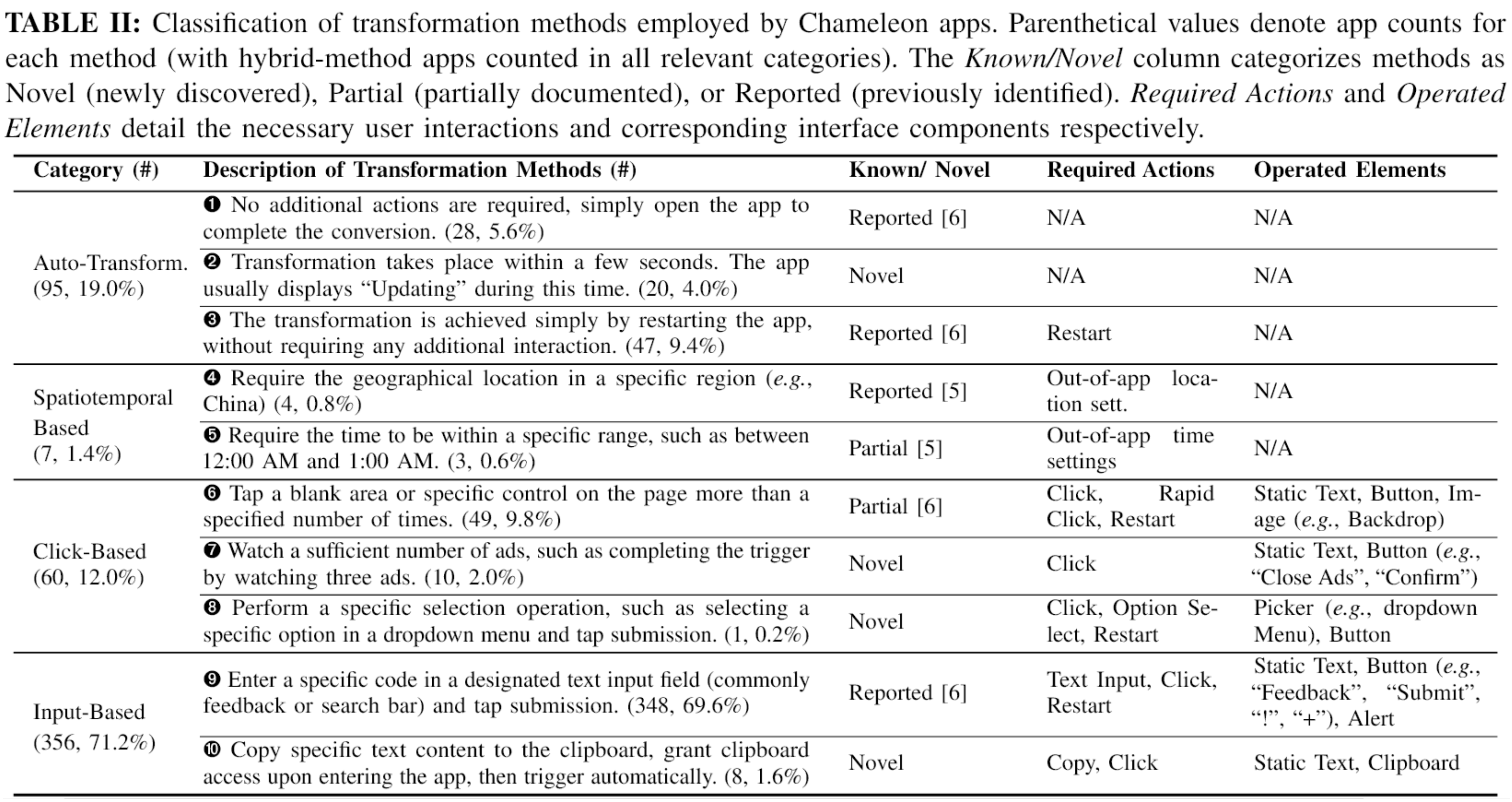


TABLE II: Classification of transformation methods employed by Chameleon apps. Parenthetical values denote app counts for each method (with hybrid-method apps counted in all relevant categories). The *Known/Novel* column categorizes methods as Novel (newly discovered), Partial (partially documented), or Reported (previously identified). *Required Actions* and *Operated Elements* detail the necessary user interactions and corresponding interface components respectively.

Category (#)	Description of Transformation Methods (#)	Known/ Novel	Required Actions	Operated Elements
Auto-Transform. (95, 19.0%)	❶ No additional actions are required, simply open the app to complete the conversion. (28, 5.6%)	Reported [6]	N/A	N/A
	❷ Transformation takes place within a few seconds. The app usually displays "Updating" during this time. (20, 4.0%)	Novel	N/A	N/A
	❸ The transformation is achieved simply by restarting the app, without requiring any additional interaction. (47, 9.4%)	Reported [6]	Restart	N/A
Spatiotemporal Based (7, 1.4%)	❹ Require the geographical location in a specific region (*e.g.*, China) (4, 0.8%)	Reported [5]	Out-of-app location sett.	N/A
	❺ Require the time to be within a specific range, such as between 12:00 AM and 1:00 AM. (3, 0.6%)	Partial [5]	Out-of-app time settings	N/A
Click-Based (60, 12.0%)	❻ Tap a blank area or specific control on the page more than a specified number of times. (49, 9.8%)	Partial [6]	Click, Rapid Click, Restart	Static Text, Button, Image (*e.g.*, Backdrop)
	❼ Watch a sufficient number of ads, such as completing the trigger by watching three ads. (10, 2.0%)	Novel	Click	Static Text, Button (*e.g.*, "Close Ads", "Confirm")
	❽ Perform a specific selection operation, such as selecting a specific option in a dropdown menu and tap submission. (1, 0.2%)	Novel	Click, Option Select, Restart	Picker (*e.g.*, dropdown Menu), Button
Input-Based (356, 71.2%)	❾ Enter a specific code in a designated text input field (commonly feedback or search bar) and tap submission. (348, 69.6%)	Reported [6]	Text Input, Click, Restart	Static Text, Button (*e.g.*, "Feedback", "Submit", "!", "+"), Alert
	❿ Copy specific text content to the clipboard, grant clipboard access upon entering the app, then trigger automatically. (8, 1.6%)	Novel	Copy, Click	Static Text, Clipboard

Preliminary Study — Characterization

➢ **Transformation Methods** (Listed in Table II):

1. Classified into **10 distinct categories**, further grouped into **4 types**.
2. **90%** methods involve **five or fewer steps**.

➢ **Disclosure of Transformation Methods**:

1. In **48.8%** of the apps analyzed, the transformation method is **disclosed within user reviews**.
2. **Example user reviews:**
   - *"useful: 520"* (**implicit** cues)
   - *"Click the exclamation mark '!' in the upper-right corner, enter '777ys,' then submit."* (**explicit** procedural guidance, found in **79 apps**)
3. **Apps** with **similar metadata** tend to use **identical transformation methods**.

➢ **Functionalities Before and After Transformation:**

1. Primarily masquerade as **puzzle games**, **habit trackers**, or **productivity tools.**
2. Illicit services include **media piracy**, **gambling** and **adult content**.

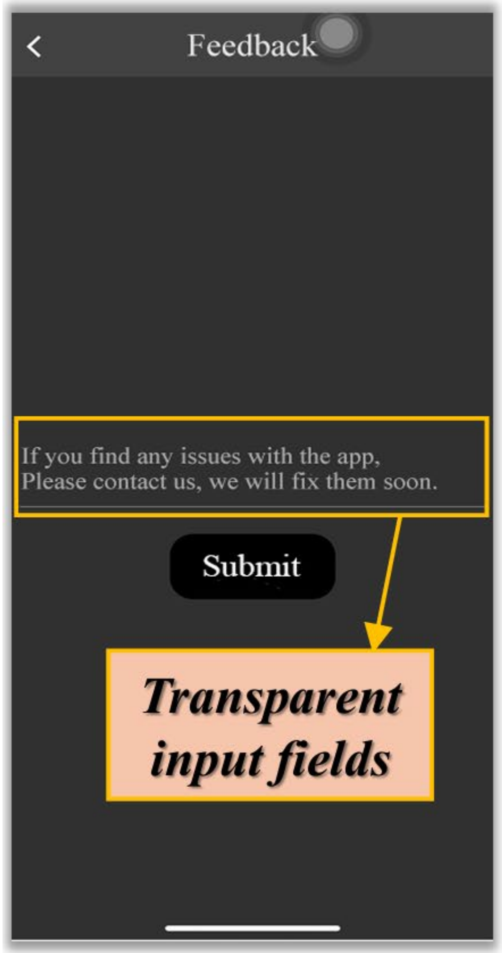
Preliminary Study — Characterization

- **Transformation Methods** (Listed in Table II):
  - ➡ Often **straightforward** and **predictable.**
- **Disclosure of Transformation Methods**:
  - ➡ Transformation methods can be reliably identified through **metadata analysis** (including user reviews, BundleIDs, and descriptions).
- **Functionalities Before and After Transformation:**
  - ➡ Successful transformation consistently leads to **distinctive UI alterations serving as visual indicators** of the illicit operational modes.

Threat Model

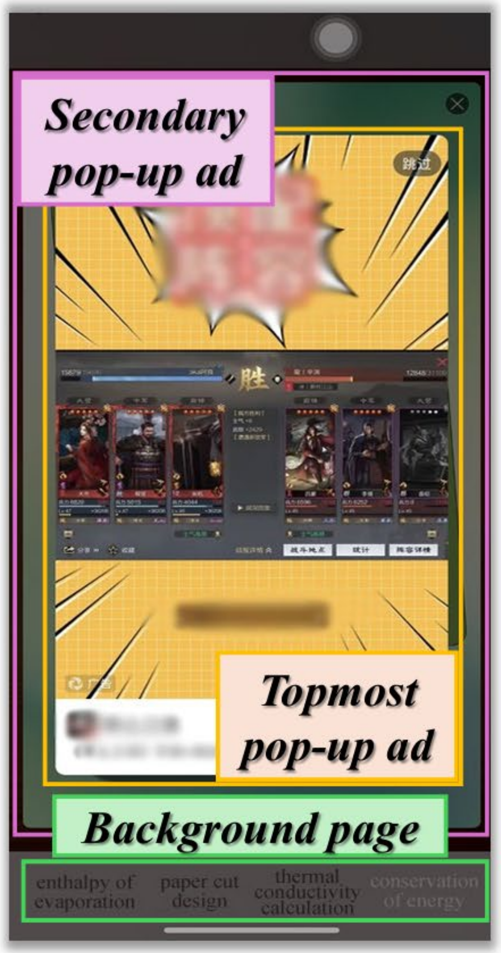
- ➢ **Insincere Developer** (**Adversary**):
  - ◆ **Develop** Chameleon apps to **bypass** iOS App Store **review** protocols.
- ➢ **When**:
  - ◆ **Successful** adversary **deployment** on the **App Store**.
- ➢ **Defender** (**Us**):
  - ◆ **Employ** automated **detection systems** to mitigate Chameleon app proliferation **post-release**.

Challenges — UI Element Recognition

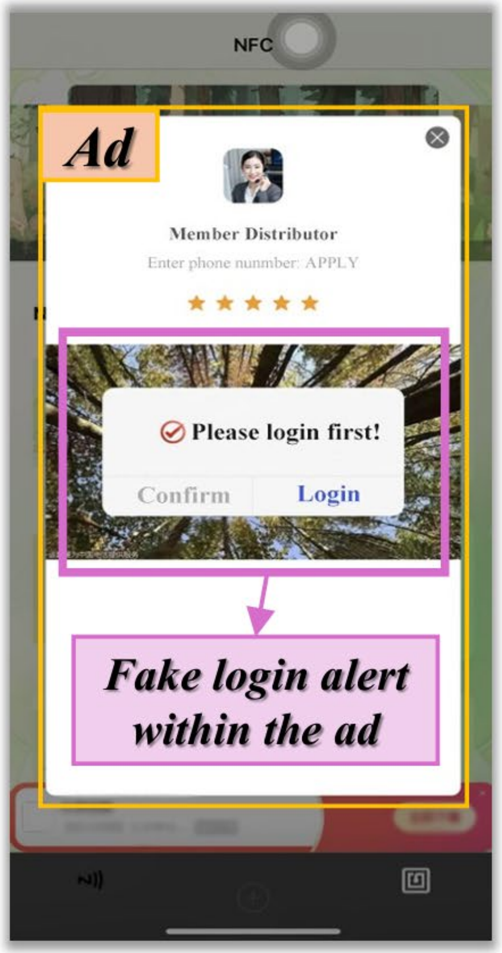


If you find any issues with the app, Please contact us, we will fix them soon.

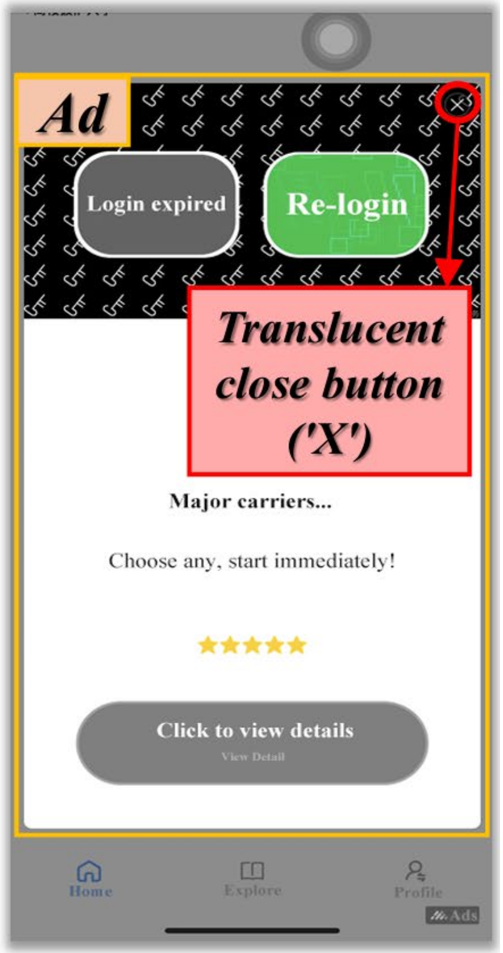
(a) Unable to recognize transparent input fields solely by vision



(b) Unable to differentiate underlying background pages from foreground overlay windows



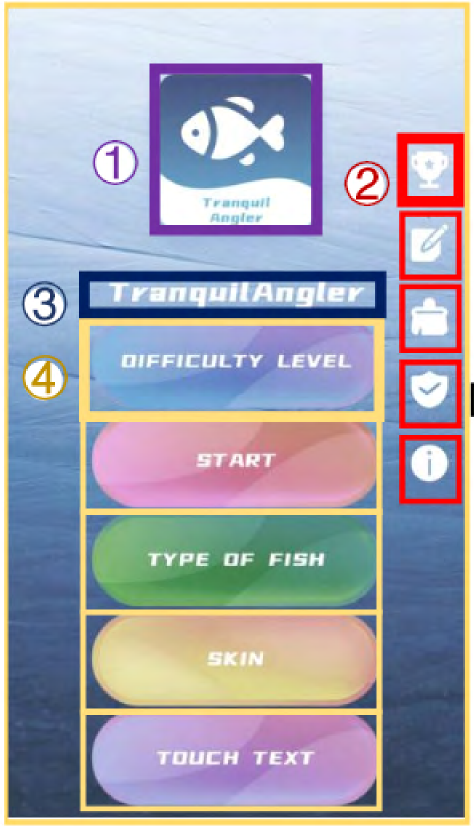
(c) Unable to resist the interference from deceptive interactive elements in pop-up ads



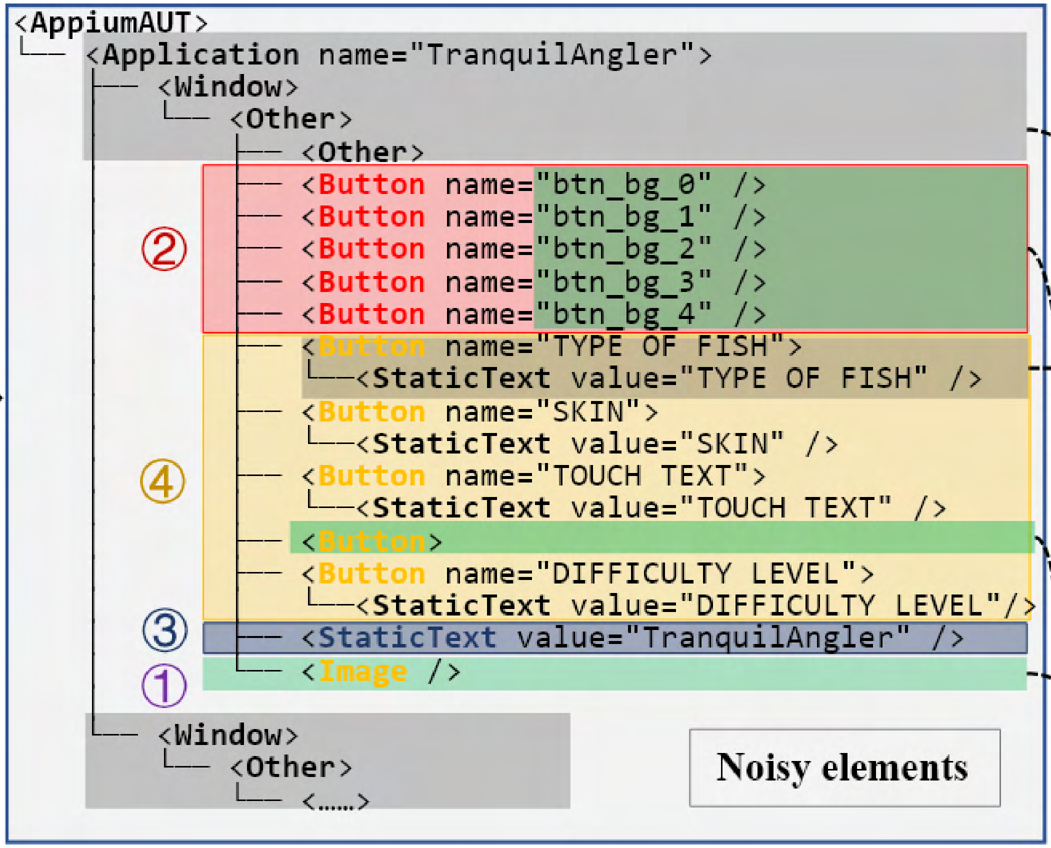
(d) Unable to detect translucent close buttons ('X') in ads

Illustrated limitations in vision-based UI Recognition.

Challenges — View Hierarchy



UI Screenshot



```
<AppiumAUT>
└── <Application name="TranquilAngler">
    ├── <Window>
    │   └── <Other>
    │       ├── <Other>
    │       ├── <Button name="btn_bg_0" />
    │       ├── <Button name="btn_bg_1" />
    │       ├── <Button name="btn_bg_2" />
    │       ├── <Button name="btn_bg_3" />
    │       ├── <Button name="btn_bg_4" />
    │       ├── <Button name="TYPE OF FISH">
    │       │   └──<StaticText value="TYPE OF FISH" />
    │       ├── <Button name="SKIN">
    │       │   └──<StaticText value="SKIN" />
    │       ├── <Button name="TOUCH TEXT">
    │       │   └──<StaticText value="TOUCH TEXT" />
    │       ├── <Button>
    │       ├── <Button name="DIFFICULTY LEVEL">
    │       │   └──<StaticText value="DIFFICULTY LEVEL"/>
    │       ├── <StaticText value="TranquilAngler" />
    │       └── <Image />
    └── <Window>
        └── <Other>
            └── <......>
```

Initial View Hierarchy (UI Tree)

Issues with View Hierarchy

(1) Noisy Elements: Excessive *<Other>* nodes w/o meaningful child nodes

(2) Redundancy: Duplicate instances of interested values (e.g., *"TYPE OF FISH"*)

(3) Semantic Deficiency: Missing semantic labels for icon buttons (②), the *START* button (in ④), and the image (①)

(4) Structural Misalignment: Discrepancy between UI tree ordering (② ④ ③ ①) and visual layout sequence(① ② ③ ④)

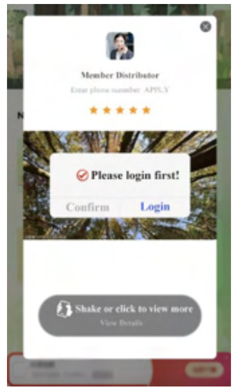
Example **UI screenshot and corresponding view hierarchy**, annotated with **identified view hierarchy issues**.

Challenges — UI Navigation

Fine-Grained UI Navigation

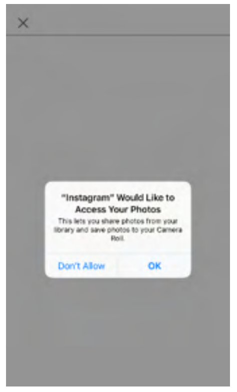
(1) Disruption of Distracting Elements

↳ e.g., **ads**, **permission requests**


Member Distributor

Enter phone number APPLY

Please login first!


"Instagram" Would Like to Access Your Photos

This lets you share photos from your library and save photos to your Camera Roll.

(2) Identification of Repetitive vs. Required Actions

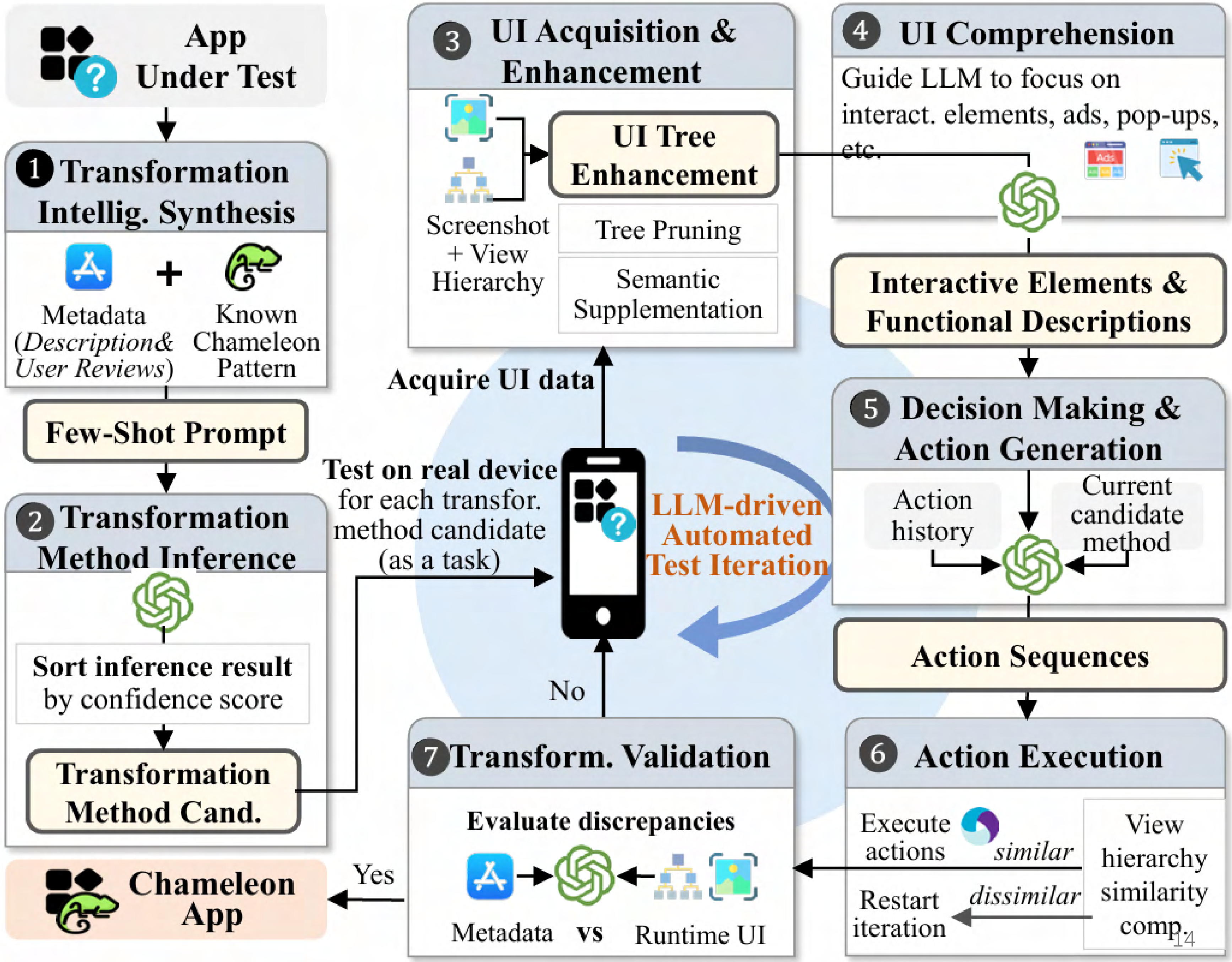
↳ e.g., **repeatedly tapping "Next"** [required]
continuously closing ads [required]
log in → register → log in → register…… [redundant]

(3) Ambiguous Task Specifications

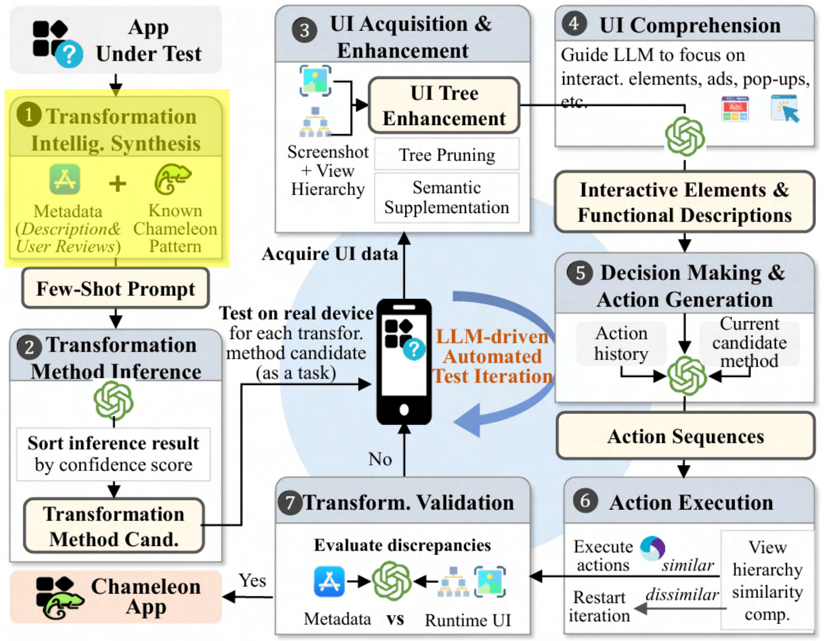
↳ e.g., **"Enter 666 to (?)"**

User review: "*666, 666, 666, 666,* …" (**input location unspecified**).

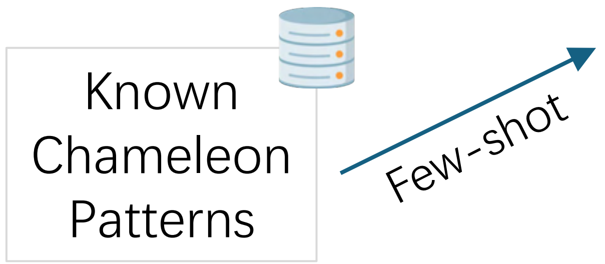
Design — Workflow of ChameleoScan

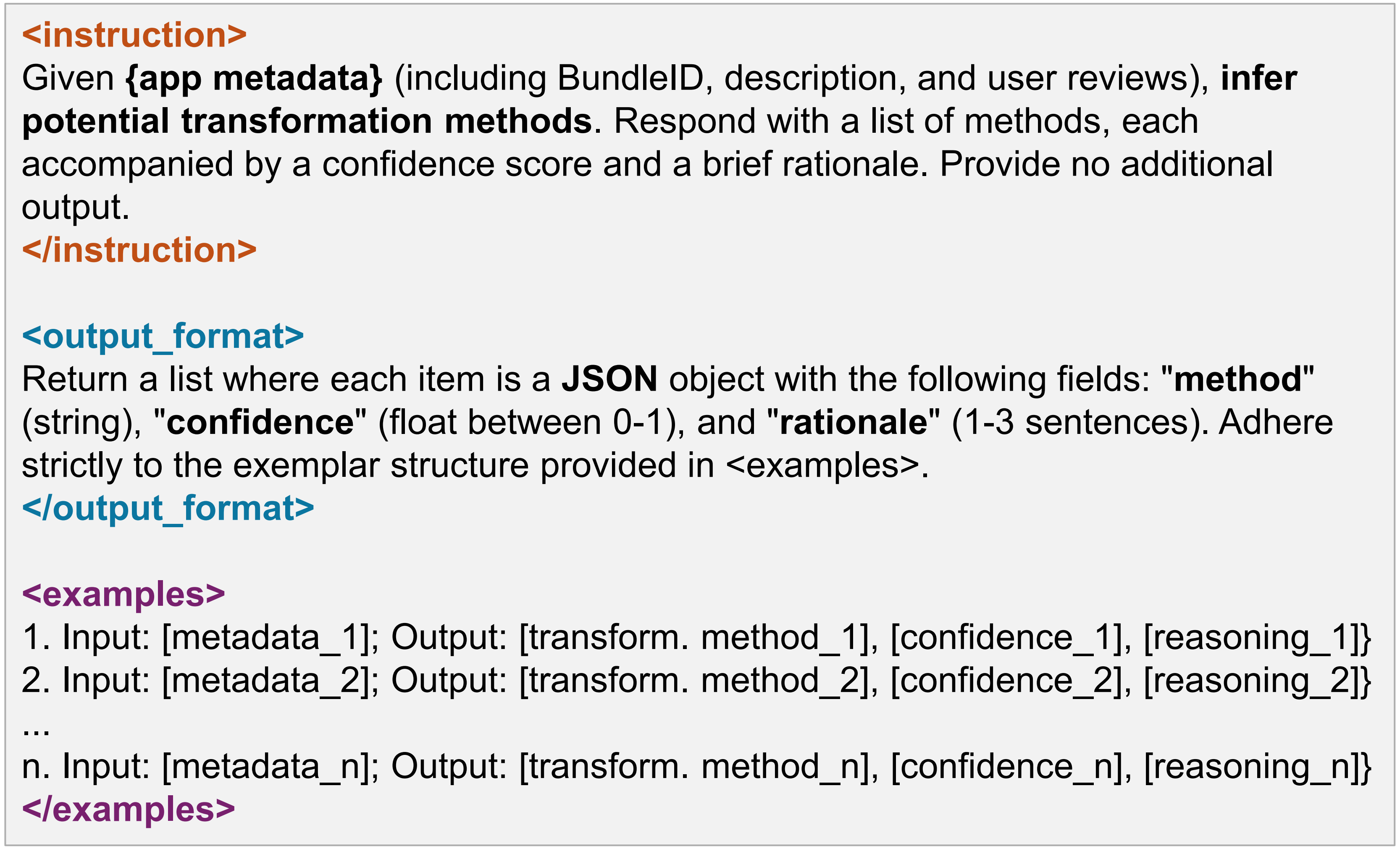

App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
+
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
Test on real device for each transfor. method candidate (as a task)
Acquire UI data
No
3 UI Acquisition & Enhancement
Screenshot + View Hierarchy
UI Tree Enhancement
Tree Pruning
Semantic Supplementation
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
LLM-driven Automated Test Iteration
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.
7 Transform. Validation
Evaluate discrepancies
Metadata
vs
Runtime UI

Design — ❶ Transformation Intelligence Synthesis



App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
3 UI Acquisition & Enhancement
UI Tree Enhancement
Screenshot + View Hierarchy
Tree Pruning
Semantic Supplementation
Acquire UI data
Test on real device for each transfor. method candidate (as a task)
LLM-driven Automated Test Iteration
No
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.
7 Transform. Validation
Evaluate discrepancies
Metadata vs Runtime UI



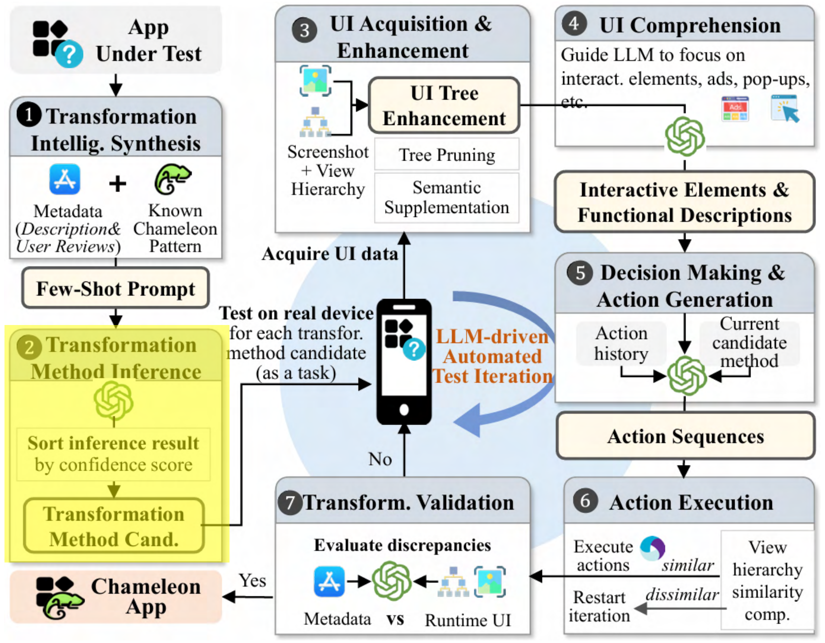

<instruction>
Given **{app metadata}** (including BundleID, description, and user reviews), **infer potential transformation methods**. Respond with a list of methods, each accompanied by a confidence score and a brief rationale. Provide no additional output.
</instruction>

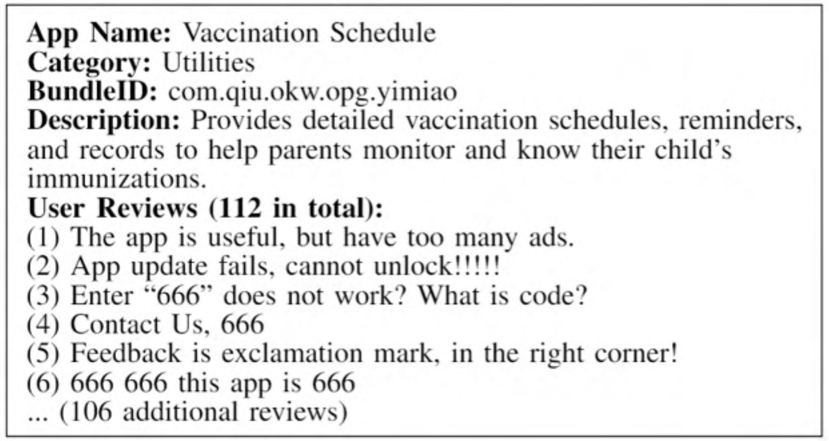
<output_format>
Return a list where each item is a **JSON** object with the following fields: "**method**" (string), "**confidence**" (float between 0-1), and "**rationale**" (1-3 sentences). Adhere strictly to the exemplar structure provided in <examples>.
</output_format>

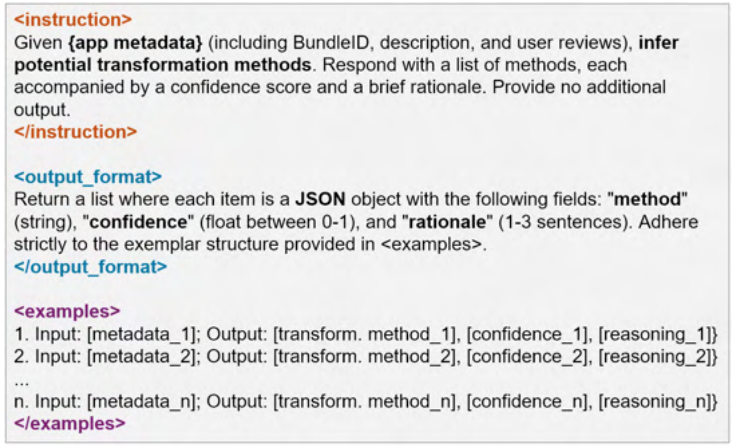
<examples>
1. Input: [metadata_1]; Output: [transform. method_1], [confidence_1], [reasoning_1]}
2. Input: [metadata_2]; Output: [transform. method_2], [confidence_2], [reasoning_2]}
...
n. Input: [metadata_n]; Output: [transform. method_n], [confidence_n], [reasoning_n]}
</examples>

Listing 1: Prompt template for transform. method inference.

Design — ❷ Transformation Method Inference


App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
+
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
3 UI Acquisition & Enhancement
UI Tree Enhancement
Screenshot + View Hierarchy
Tree Pruning
Semantic Supplementation
Acquire UI data
Test on real device for each transfor. method candidate (as a task)
LLM-driven Automated Test Iteration
No
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.
7 Transform. Validation
Evaluate discrepancies
Metadata
vs
Runtime UI


App Name: Vaccination Schedule
Category: Utilities
BundleID: com.qiu.okw.opg.yimiao
Description: Provides detailed vaccination schedules, reminders, and records to help parents monitor and know their child's immunizations.
User Reviews (112 in total):
(1) The app is useful, but have too many ads.
(2) App update fails, cannot unlock!!!!!
(3) Enter "666" does not work? What is code?
(4) Contact Us, 666
(5) Feedback is exclamation mark, in the right corner!
(6) 666 666 this app is 666
... (106 additional reviews)


<instruction>
Given **{app metadata}** (including BundleID, description, and user reviews), **infer potential transformation methods**. Respond with a list of methods, each accompanied by a confidence score and a brief rationale. Provide no additional output.
</instruction>

<output_format>
Return a list where each item is a **JSON** object with the following fields: "**method**" (string), "**confidence**" (float between 0-1), and "**rationale**" (1-3 sentences). Adhere strictly to the exemplar structure provided in <examples>.
</output_format>

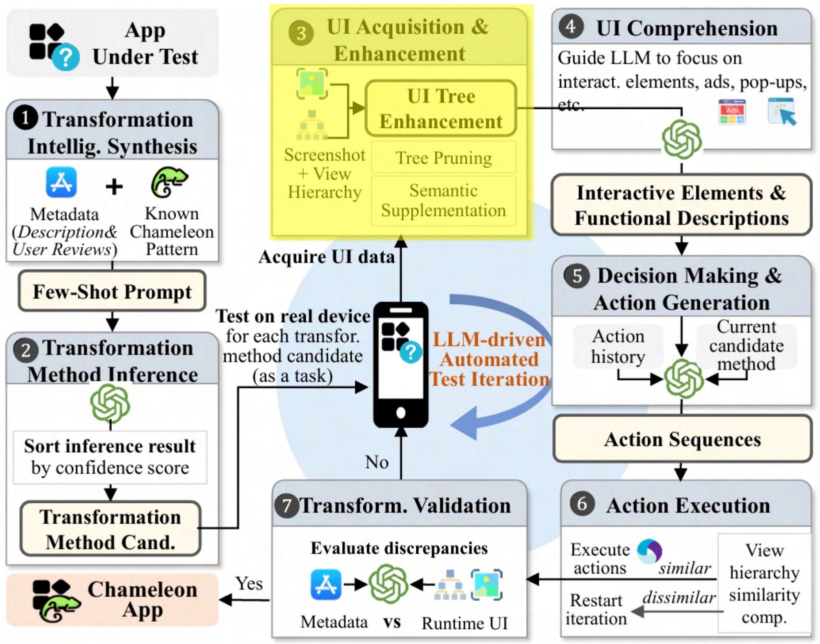
<examples>
1. Input: [metadata_1]; Output: [transform. method_1], [confidence_1], [reasoning_1]}
2. Input: [metadata_2]; Output: [transform. method_2], [confidence_2], [reasoning_2]}
...
n. Input: [metadata_n]; Output: [transform. method_n], [confidence_n], [reasoning_n]}
</examples>

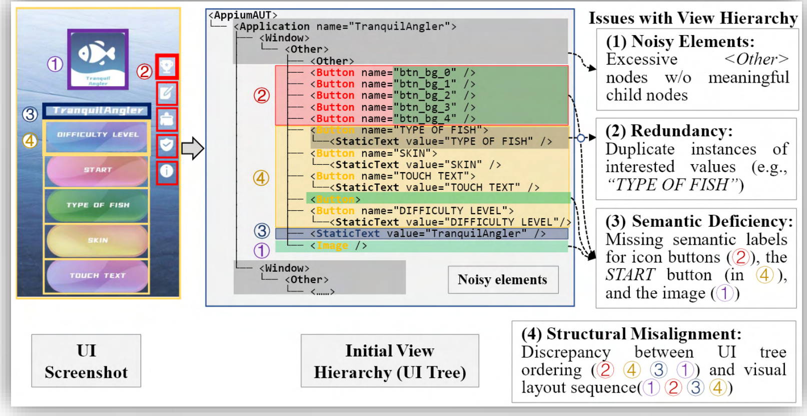
App Metadata + Few-shot Prompt

Output:
A limited set (typically n=2 or n=3) of tuples structured as {**transformation method**, **confidence score**, **reasoning**}

Examples:
(1) Click "Contact Us", input "666" & submit feedback (0.9) - [reasoning]
(2) Restart the app and wait for update for three times (0.4) – [reasoning]
(3) Watch ads before clicking the "Close Ads" button (0.2) – [reasoning]

Design — ❸ UI Data Acquisition and Enhancement


App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
+
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
3 UI Acquisition & Enhancement
Screenshot + View Hierarchy
UI Tree Enhancement
Tree Pruning
Semantic Supplementation
Acquire UI data
Test on real device for each transfor. method candidate (as a task)
LLM-driven Automated Test Iteration
No
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.
7 Transform. Validation
Evaluate discrepancies
Metadata vs Runtime UI



```
<AppiumAUT>
└── <Application name="TranquilAngler">
    ├── <Window>
    │   └── <Other>
    │       ├── <Other>
    │       ├── <Button name="btn_bg_0" />
    │       ├── <Button name="btn_bg_1" />
    │       ├── <Button name="btn_bg_2" />
    │       ├── <Button name="btn_bg_3" />
    │       ├── <Button name="btn_bg_4" />
    │       ├── <Button name="TYPE OF FISH">
    │       │   └──<StaticText value="TYPE OF FISH" />
    │       ├── <Button name="SKIN">
    │       │   └──<StaticText value="SKIN" />
    │       ├── <Button name="TOUCH TEXT">
    │       │   └──<StaticText value="TOUCH TEXT" />
    │       ├── <Button>
    │       ├── <Button name="DIFFICULTY LEVEL">
    │       │   └──<StaticText value="DIFFICULTY LEVEL"/>
    │       ├── <StaticText value="TranquilAngler" />
    │       └── <Image />
    └── <Window>
        └── <Other>
            └── <......>
```

Issues with View Hierarchy

(1) Noisy Elements: Excessive *<Other>* nodes w/o meaningful child nodes

(2) Redundancy: Duplicate instances of interested values (e.g., *"TYPE OF FISH"*)

(3) Semantic Deficiency: Missing semantic labels for icon buttons (②), the *START* button (in ④), and the image (①)

(4) Structural Misalignment: Discrepancy between UI tree ordering (② ④ ③ ①) and visual layout sequence(① ② ③ ④)

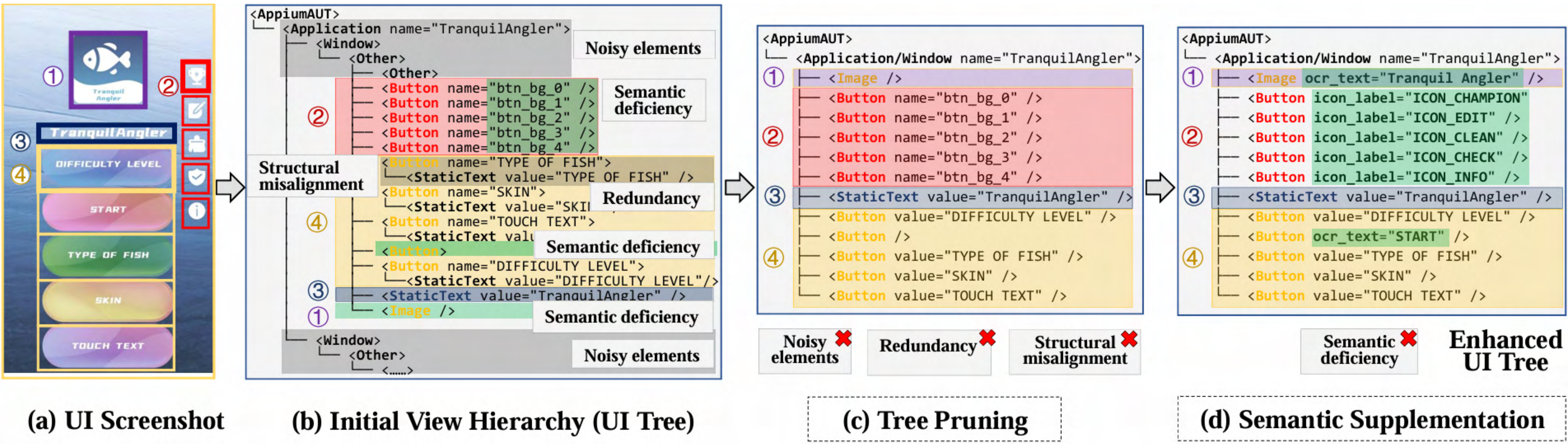
To solve challenges brought by **UI Element Recognition** and **View Hierarchy**:

1. **Tree Pruning:**
   - Removing noisy elements
   - Reducing redundancy
   - Correcting structural misalignment

2. **Semantic Supplementation**:
   - For Icons, generate semantic annotations
   - For Images, extract bounded texts

Design — ❸ UI Data Acquisition and Enhancement

Figure: View hierarchy (UI tree) enhancement procedure demonstrated using the “Tranquil Angler” app.



```
<AppiumAUT>
└── <Application name="TranquilAngler">
    ├── <Window>
    │   └── <Other>
    │       ├── <Other>
    │       ├── <Button name="btn_bg_0" />
    │       ├── <Button name="btn_bg_1" />
    │       ├── <Button name="btn_bg_2" />
    │       ├── <Button name="btn_bg_3" />
    │       ├── <Button name="btn_bg_4" />
    │       ├── <Button name="TYPE OF FISH">
    │       │   └─<StaticText value="TYPE OF FISH" />
    │       ├── <Button name="SKIN">
    │       │   └─<StaticText value="SKIN
    │       ├── <Button name="TOUCH TEXT">
    │       │   └─<StaticText valu
    │       ├── <Button>
    │       ├── <Button name="DIFFICULTY LEVEL">
    │       │   └─<StaticText value="DIFFICULTY LEVEL"/>
    │       ├── <StaticText value="TranquilAngler" />
    │       └── <Image />
    └── <Window>
        └── <Other>
            └── <......>
```

```
<AppiumAUT>
└── <Application/Window name="TranquilAngler">
    ├── <Image />
    ├── <Button name="btn_bg_0" />
    ├── <Button name="btn_bg_1" />
    ├── <Button name="btn_bg_2" />
    ├── <Button name="btn_bg_3" />
    ├── <Button name="btn_bg_4" />
    ├── <StaticText value="TranquilAngler" />
    ├── <Button value="DIFFICULTY LEVEL" />
    ├── <Button />
    ├── <Button value="TYPE OF FISH" />
    ├── <Button value="SKIN" />
    └── <Button value="TOUCH TEXT" />
```

```
<AppiumAUT>
└── <Application/Window name="TranquilAngler">
    ├── <Image ocr_text="Tranquil Angler" />
    ├── <Button icon_label="ICON_CHAMPION"
    ├── <Button icon_label="ICON_EDIT" />
    ├── <Button icon_label="ICON_CLEAN" />
    ├── <Button icon_label="ICON_CHECK" />
    ├── <Button icon_label="ICON_INFO" />
    ├── <StaticText value="TranquilAngler" />
    ├── <Button value="DIFFICULTY LEVEL" />
    ├── <Button ocr_text="START" />
    ├── <Button value="TYPE OF FISH" />
    ├── <Button value="SKIN" />
    └── <Button value="TOUCH TEXT" />
```

(a) UI Screenshot (b) Initial View Hierarchy (UI Tree) (c) Tree Pruning (d) Semantic Supplementation

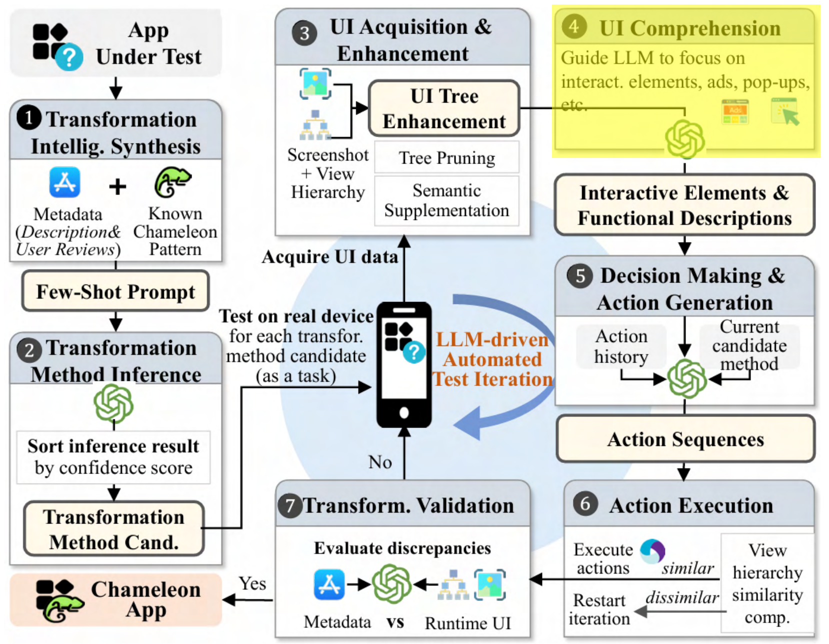
Tree Pruning

1. **Remove noisy elements** (auxiliary nodes, empty views, and off-screen elements).
2. **Reduce redundancy** by merging similar parent-child nodes and keeping only **distinct** attribute values.
3. **Fix structural misalignment** by reordering nodes using their absolute **grid coordinates**.

Semantic Supplem-entation

1. **For icons**, we train an **EfficientNet-B0** classifier on the **RICO** Semantics dataset to **predict semantic labels**.
2. **For images**, we apply **PaddleOCR** to **extract text** and assign it to the **most precise node** (text-containing with minimal area).

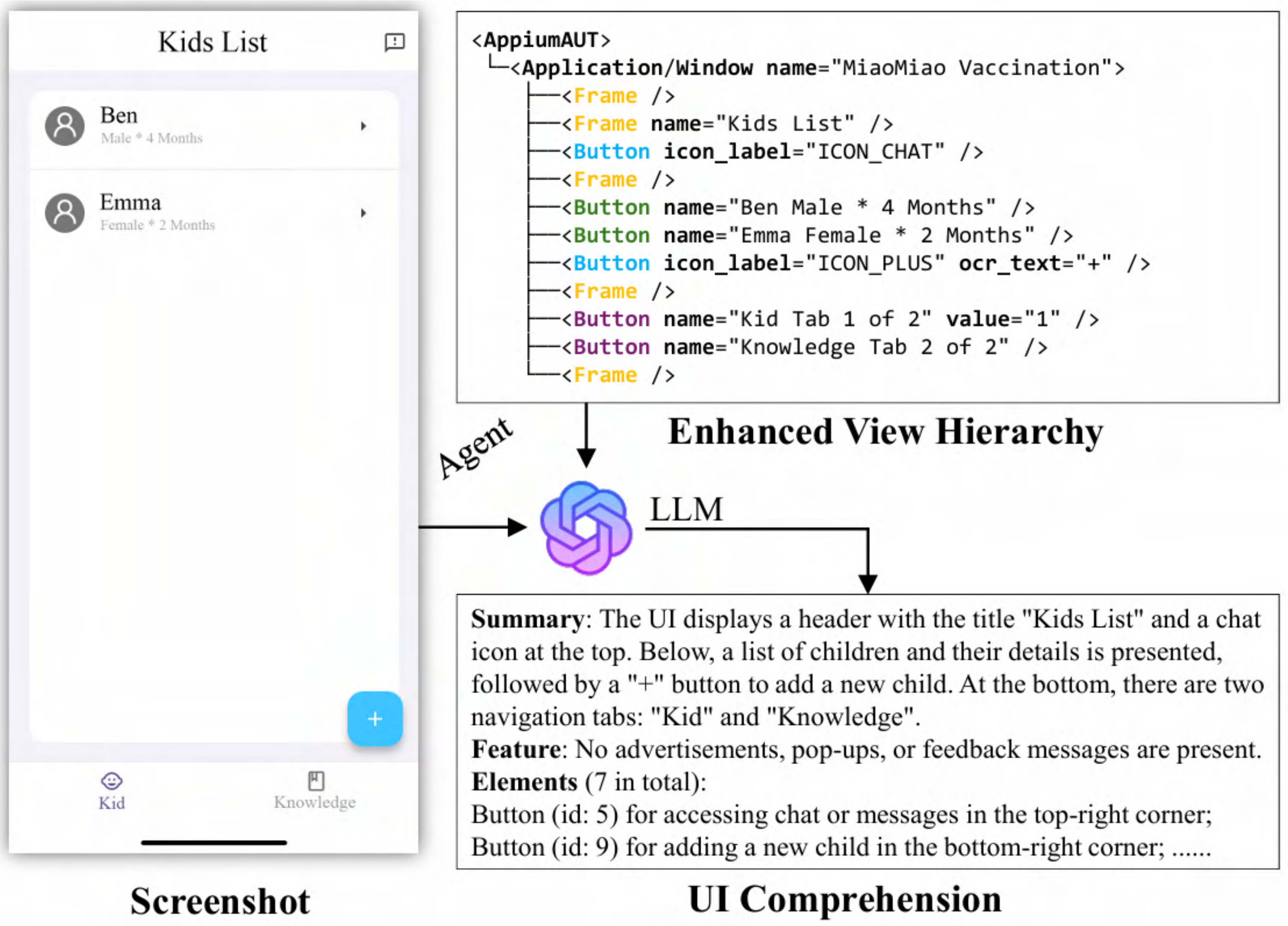
Design — ❹ UI Comprehension



App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
+
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
3 UI Acquisition & Enhancement
Screenshot + View Hierarchy
UI Tree Enhancement
Tree Pruning
Semantic Supplementation
Acquire UI data
Test on real device for each transfor. method candidate (as a task)
LLM-driven Automated Test Iteration
No
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.
7 Transform. Validation
Evaluate discrepancies
Metadata vs Runtime UI

<instruction>
Given the {**view hierarchy in XML**} and a {**runtime screenshot**}, perform comprehensive UI analysis by: (1) **establishing visual correspondence** between hierarchy elements and screenshot regions while filtering non-matching or obscured components; (2) **classifying** all UI elements as either **non-interactive** (e.g., labels, static content) or **interactive** (e.g., buttons, clickable icons, input fields). **Pay special attention to** promotional content, transient dialogs, and system messages, with documentation of their dismissal mechanisms. Present the final analysis as a **structured** inventory of UI elements without additional commentary.
</instruction>
<output_format>
Return a list where each item is a JSON object representing a single UI element with the following fields: "**type**" (string), "**state**" (string), "**text**" (string), and "**description**".
</output_format>

Listing 2: Prompt template for UI comprehension.



```
<AppiumAUT>
└<Application/Window name="MiaoMiao Vaccination">
   ├<Frame />
   ├<Frame name="Kids List" />
   ├<Button icon_label="ICON_CHAT" />
   ├<Frame />
   ├<Button name="Ben Male * 4 Months" />
   ├<Button name="Emma Female * 2 Months" />
   ├<Button icon_label="ICON_PLUS" ocr_text="+" />
   ├<Frame />
   ├<Button name="Kid Tab 1 of 2" value="1" />
   ├<Button name="Knowledge Tab 2 of 2" />
   └<Frame />
```

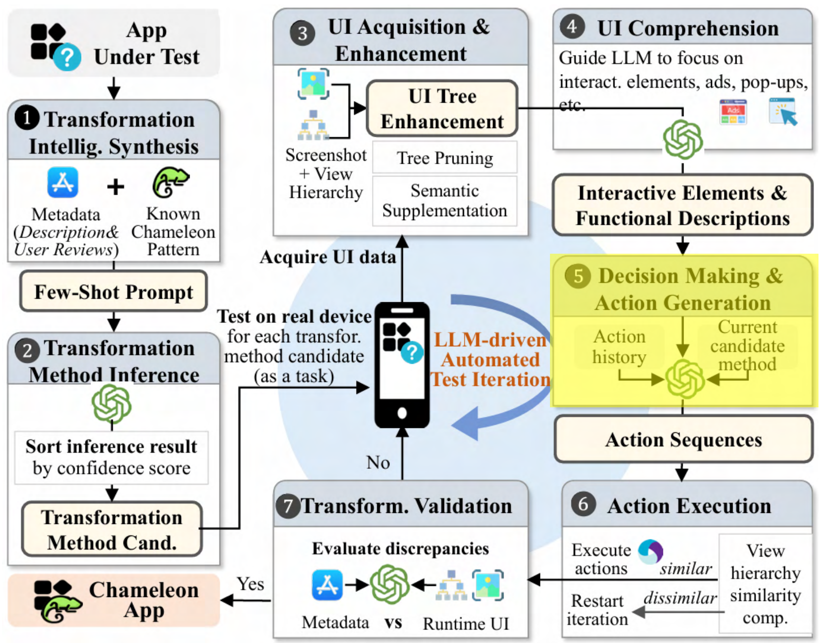
Enhanced View Hierarchy

Summary: The UI displays a header with the title "Kids List" and a chat icon at the top. Below, a list of children and their details is presented, followed by a "+" button to add a new child. At the bottom, there are two navigation tabs: "Kid" and "Knowledge".
Feature: No advertisements, pop-ups, or feedback messages are present.
Elements (7 in total):
Button (id: 5) for accessing chat or messages in the top-right corner;
Button (id: 9) for adding a new child in the bottom-right corner; ......

Screenshot **UI Comprehension**

Fig. 7: The main **interface** of *Vaccination Schedule*, its enhanced **view hierarchy**, and the **output** of UI comprehension.

Design — ❺ Decision Making & Action Generation



App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
3 UI Acquisition & Enhancement
Screenshot + View Hierarchy
UI Tree Enhancement
Tree Pruning
Semantic Supplementation
Acquire UI data
Test on real device for each transfor. method candidate (as a task)
LLM-driven Automated Test Iteration
No
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.
7 Transform. Validation
Evaluate discrepancies
Metadata vs Runtime UI

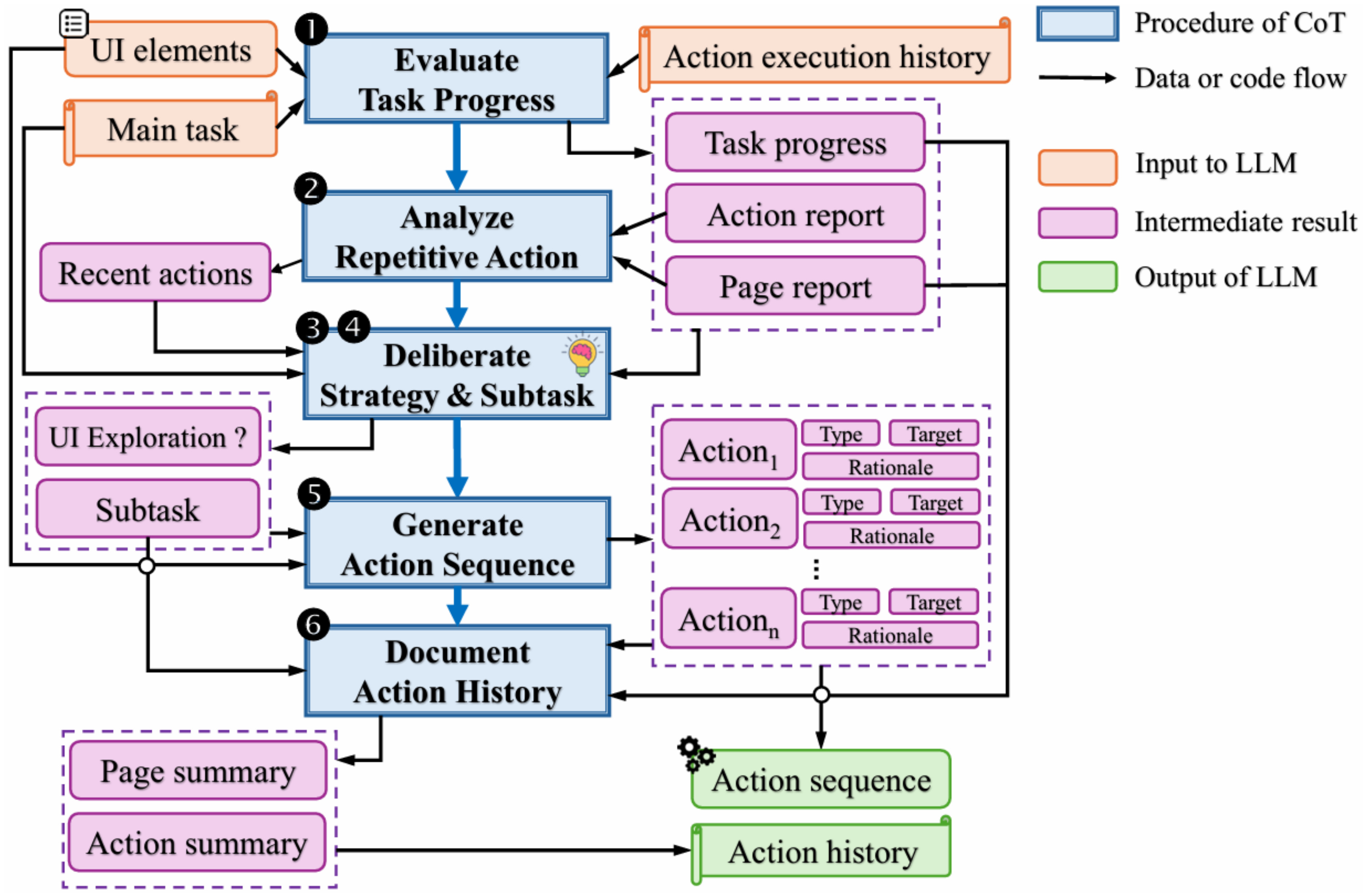
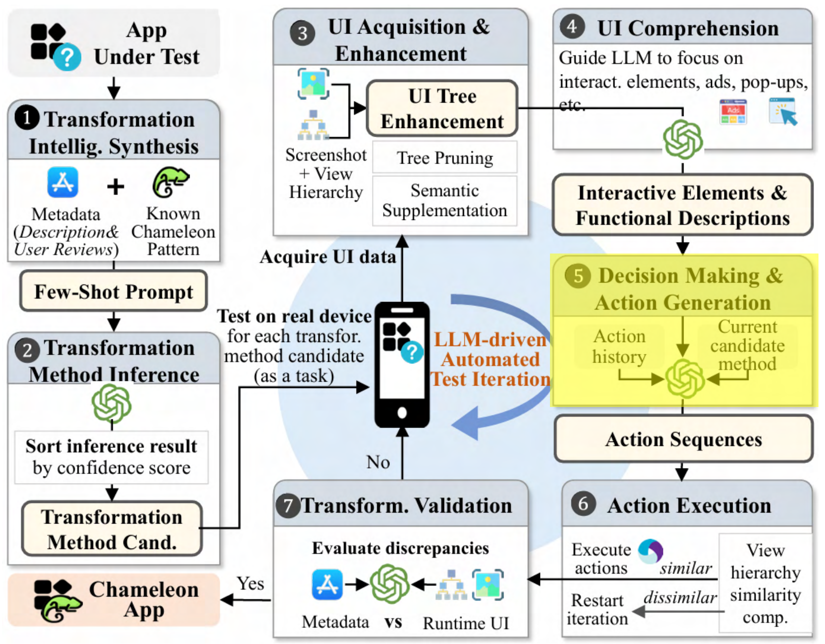


Fig. 8: **Chain-of-thought** reasoning process for decision making and action sequence generation.

Challenges of UI Navigation:

1. Disruption of Distracting Elements
2. Identification of Repetitive vs. Required Actions
3. Ambiguous Task Specifications

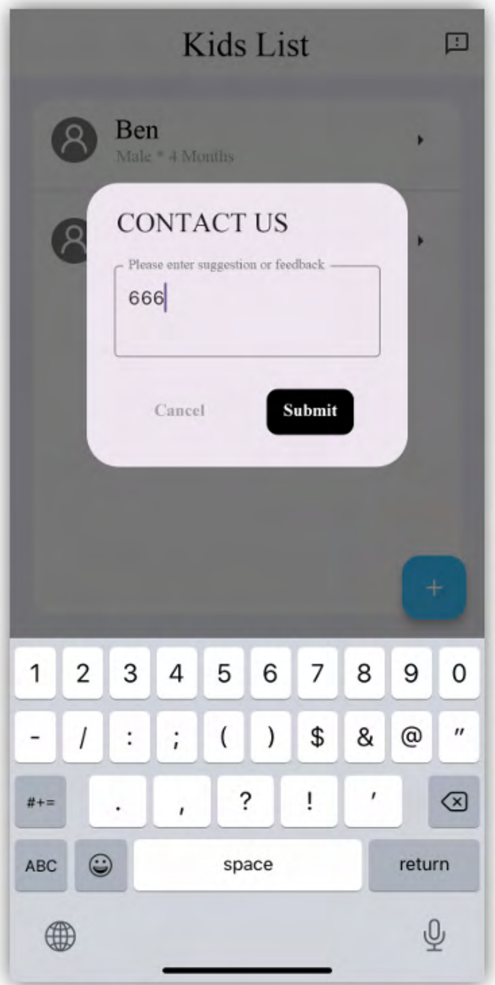
Design — ❺ Decision Making & Action Generation



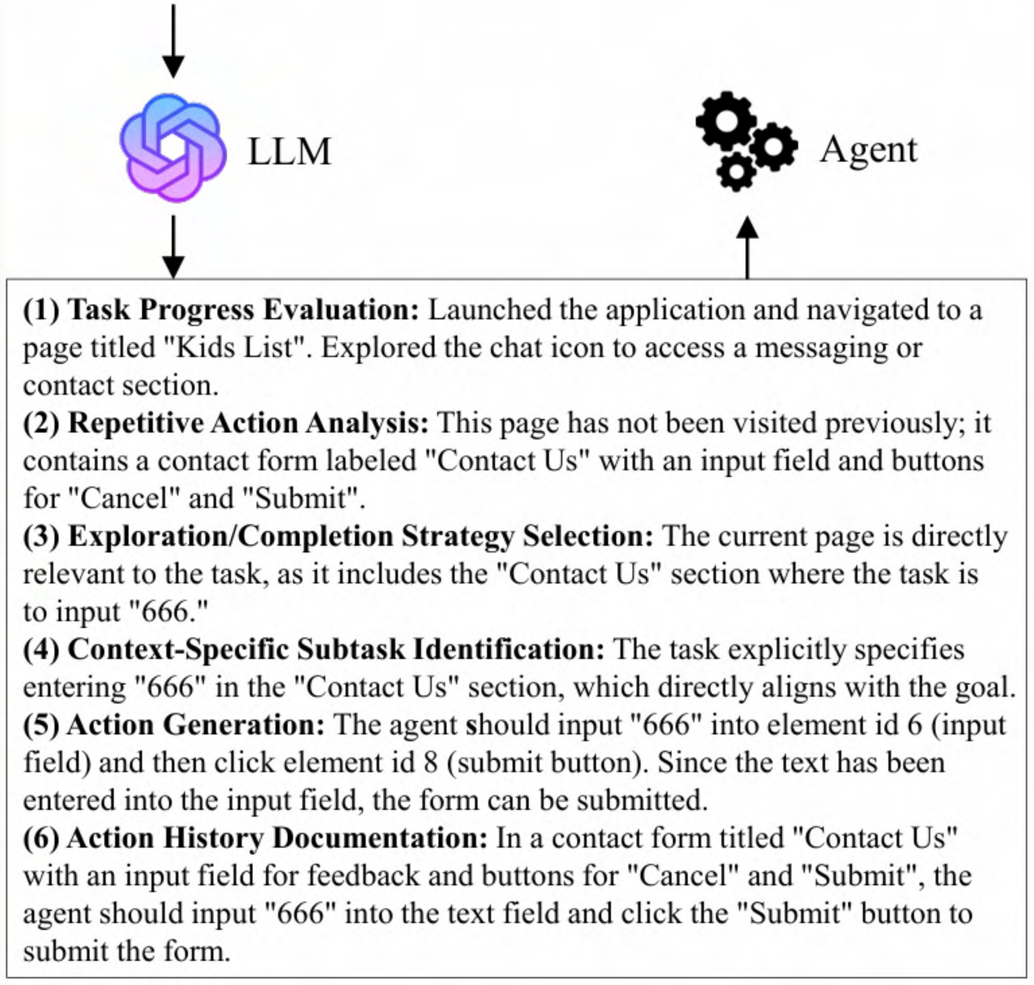
App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
3 UI Acquisition & Enhancement
Screenshot + View Hierarchy
UI Tree Enhancement
Tree Pruning
Semantic Supplementation
Acquire UI data
Test on real device for each transfor. method candidate (as a task)
LLM-driven Automated Test Iteration
No
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.
7 Transform. Validation
Evaluate discrepancies
Metadata vs Runtime UI

<instruction>
Given the {**current UI elements**}, {**action execution history**}, and {**primary task**}, employ structured reasoning to: (1) **identify the most appropriate immediate sub-task**, and (2) **formulate a coherent, goal-oriented action sequence**. Prioritize handling **urgent UI components** (e.g., advertisements, pop-up dialogs), **avoid redundant operations**, and ensure all proposed actions maintain **contextual relevance**, **operational feasibility**, and **explicit justification**. Present the analysis as **structured** output without additional commentary.
</instruction>
<output_format>
Return a single JSON object containing the following fields: "**summary**" (string), "**history_analysis**" (string), "**subtask**" (string), and "**action_plan**" (a list of JSON objects, each containing "**action**" and "**rationale**" strings).
</output_format>

Listing 3: Prompt template for decision making and action sequence generation.



Screenshot

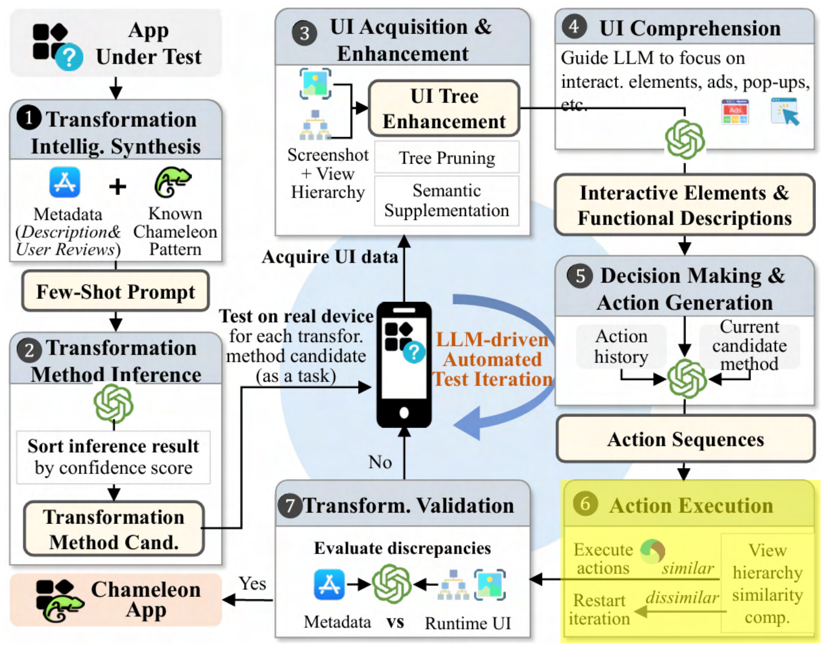


(1) Task Progress Evaluation: Launched the application and navigated to a page titled "Kids List". Explored the chat icon to access a messaging or contact section.
(2) Repetitive Action Analysis: This page has not been visited previously; it contains a contact form labeled "Contact Us" with an input field and buttons for "Cancel" and "Submit".
(3) Exploration/Completion Strategy Selection: The current page is directly relevant to the task, as it includes the "Contact Us" section where the task is to input "666."
(4) Context-Specific Subtask Identification: The task explicitly specifies entering "666" in the "Contact Us" section, which directly aligns with the goal.
(5) Action Generation: The agent should input "666" into element id 6 (input field) and then click element id 8 (submit button). Since the text has been entered into the input field, the form can be submitted.
(6) Action History Documentation: In a contact form titled "Contact Us" with an input field for feedback and buttons for "Cancel" and "Submit", the agent should input "666" into the text field and click the "Submit" button to submit the form.

Decision Making and Action Sequence Generation

Fig. 9: The Decision Making and Action Sequence Generation module's **output** for *Vaccination Schedule*'s current UI.

Design — ❻ Action Execution



App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
No
Test on real device for each transfor. method candidate (as a task)
LLM-driven Automated Test Iteration
Acquire UI data
3 UI Acquisition & Enhancement
Screenshot + View Hierarchy
UI Tree Enhancement
Tree Pruning
Semantic Supplementation
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.
7 Transform. Validation
Evaluate discrepancies
Metadata vs Runtime UI

② **Pre-check** whether the **UI is still valid** before execution:

➡ mitigate transient components (ads/pop-ups)

(1) Allow dynamic content in fixed-position widgets:
Tree edit distance (of view hierarchy) < 3

or

(2) Allow structural changes in visually stable screens:
Jaccard similarity (of all text labels) > 0.8

①

Action Sequence

Action:	Parameter:	
Click	Target, Count	Interact
Input	Target, Text	Interact
Restart	N/A	Interact
Copy	Text	Interact
Complete	N/A	Control
Null	N/A	Control

③ Known UI Patterns

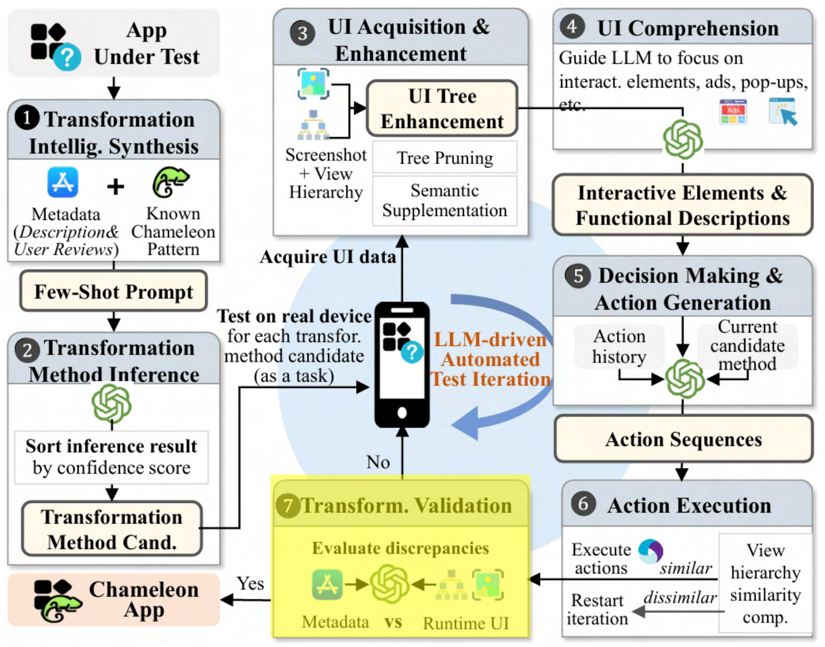
e.g., permission requests, known ads framework

Close Immediately

④ Recovery Protocols

e.g., app foregrounding, keyboard dismissal

Design — ❼ Transformation Validation


App Under Test
1 Transformation Intellig. Synthesis
Metadata (Description& User Reviews)
+
Known Chameleon Pattern
Few-Shot Prompt
2 Transformation Method Inference
Sort inference result by confidence score
Transformation Method Cand.
Chameleon App
Yes
3 UI Acquisition & Enhancement
Screenshot + View Hierarchy
UI Tree Enhancement
Tree Pruning
Semantic Supplementation
Acquire UI data
Test on real device for each transfor. method candidate (as a task)
LLM-driven Automated Test Iteration
No
7 Transform. Validation
Evaluate discrepancies
Metadata vs Runtime UI
4 UI Comprehension
Guide LLM to focus on interact. elements, ads, pop-ups, etc.
Interactive Elements & Functional Descriptions
5 Decision Making & Action Generation
Action history
Current candidate method
Action Sequences
6 Action Execution
Execute actions
similar
dissimilar
Restart iteration
View hierarchy similarity comp.

Structured Four-Phase Analysis

(1) Functionality Consistency Analysis

(2) Non-Indicative Element Filtering

(3) Evidence-Based Reasoning

(4) Confidence-Based Scoring

<instruction>
Given the {**runtime behavior**} and {**app metadata**}, determine whether the observed app runtime behavior **clearly contradicts or deviates** from the app's **declared functionality**. Base your assessment exclusively **on visible UI evidence**, citing specific **textual or visual indicators** to support your conclusions. Additionally, flag instances where **advertisements or transient elements** substantially **obstruct** interface evaluation. Provide no additional output.
</instruction>
<output_format>
Return a list containing a single **JSON** object with the following fields: "**determination**" (string), "**confidence**" (float between 0-1), and "**rationale**" (1-3 sentences). Adhere strictly to the exemplar structure provided in <examples>.
</output_format>
<examples>
1. Input: [runtime behavor_1], [metadata_1]; Output: [determination_1], [confidence_1], [reasoning_1]}
2. Input: [runtime behavor_2], [metadata_2]; Output: [determination_2], [confidence_2], [reasoning_2]}
...
n. Input: [runtime behavor_n], [metadata_n]; Output: [determination_n], [confidence_n], [reasoning_n]}
</examples>

Listing 4: Prompt template for transformation validation.

Evaluation — **Dataset and RQs**

Known dataset: **234** previously collected **chameleon apps + 233** selected **benign apps**.
Unknown dataset: **1,644 real App Store apps** collected **over a month** (previously **unseen**).

RQ1: Performance on KNOWN Dataset

→ **Precision; Recall; Accuracy; Average Round.**

RQ2: Task-wise Performance

→ (1) **Transformation Method Inference**; (2) **UI Element Recognition and Comprehension**; (3) **Interference Handling** and **Dynamic UI Management.**

RQ3: Performance on UNKNOWN Dataset

→ **Precision**; **Average Round**; Resource and **Cost Analysis**; **Insights**.

RQ4: Comparison with Chameleon-Hunter and Mask-Catcher

Evaluation — RQ1: Perf. on KNOWN Dataset

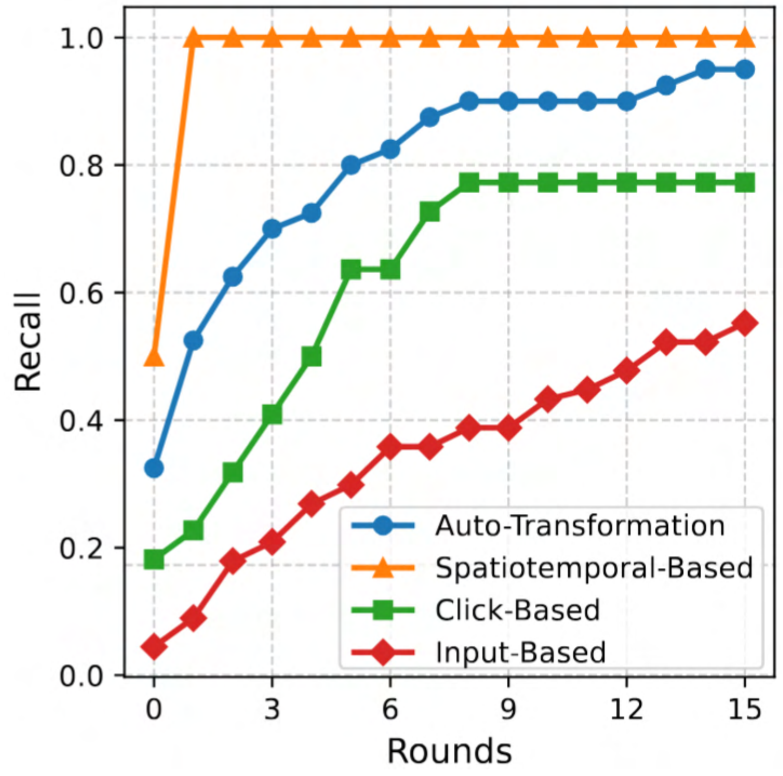
Table IV: ChameleoScan's **detection efficacy** on KNOWN

Type	AR	Recall (%)	Recall′ (%)
Auto-Transformation	2.58	95.00	100.00
Spatiotemporal-Based	0.50	100.00	100.00
Click-Based	2.81	77.27	100.00
Input-Based	4.18	55.22	92.50
Total	**3.45**	**71.76**	**96.91**

Precision: 100%
Recall': 96.91%

Near expert-level performance

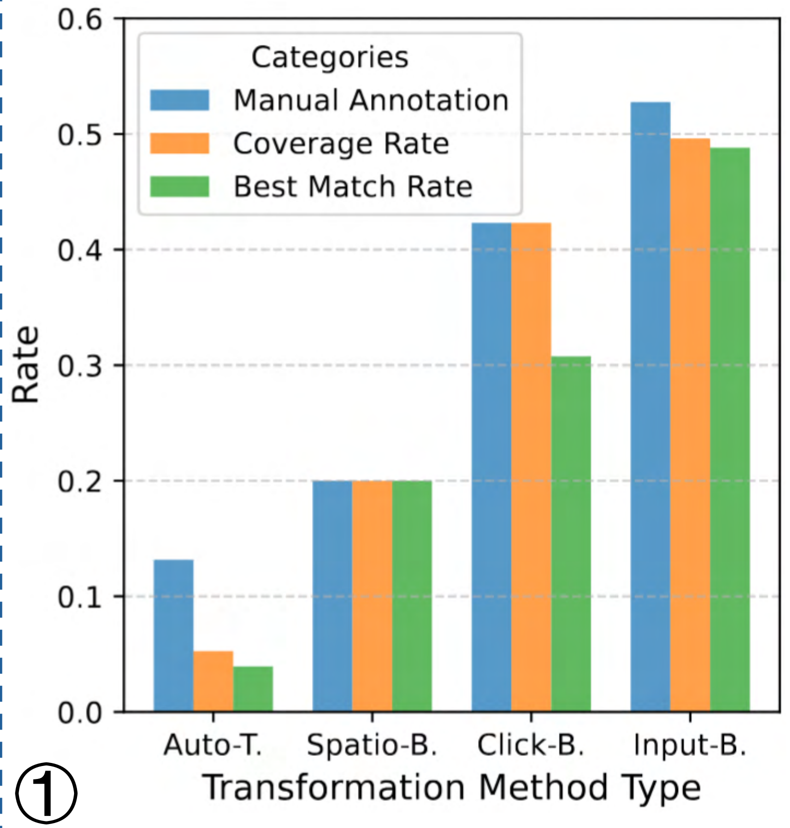
Fig. 10: (a) **recall curves** across interaction rounds by Chameleon app type



2.26 rounds per app:
chameleon: 3.45 rounds
benign: 1.59 rounds

Evaluation — RQ2: Task-wise Performance

Fig. 10: (b) accuracy of **transformation method inference** by app type



①

② **Table V**: Component Impact Analysis for **UI Element Recognition and Comprehension**: Evaluation of **tree pruning** (*Pruning*), **image semantic augmentation** (*SSimg*), and **icon semantic enhancement** (*SSicon*) on **recognition precision** (P), **recall** (R), and **accuracy** (Acc) across **927** XML formatted interface pages from the **KNOWN** dataset.

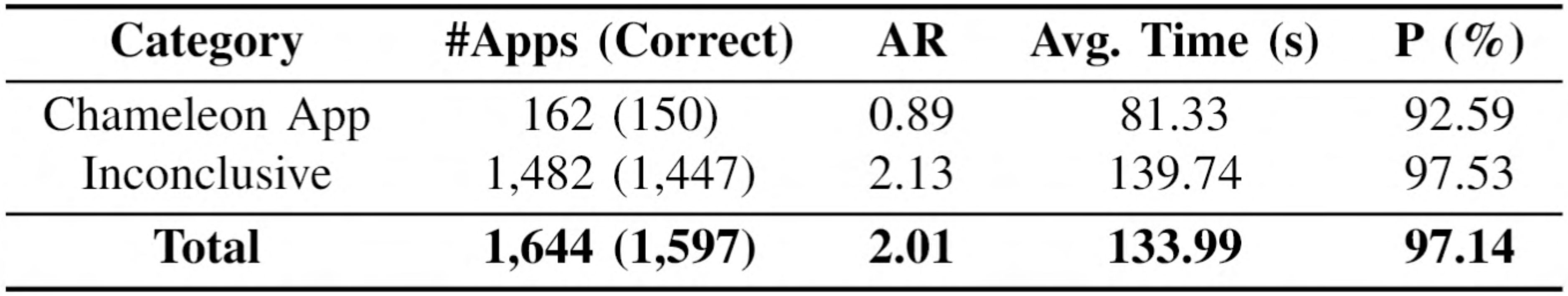
Modules	UI Recognition			UI Comprehension
	P (%)	R (%)	Acc (%)	Acc (%)
{}	85.50	63.21	74.19	87.26
{ *Pruning* }	87.04	65.45	76.02	87.10
{ *Pruning*, *SS_img* }	88.42	68.26	77.58	89.39
{ *Pruning*, *SS_icon* }	87.84	66.38	77.05	91.85
{ *Pruning*, *SS_img*, *SS_icon* }	**89.36**	**70.39**	**78.94**	**92.54**

③ **Interference Handling and Dynamic UI Management.**

- **4,415 UI pages annotated: 19.21%** contain ads/pop-ups (39.19% of apps).
- **Ads/pop-ups detection**: Precision 96.70%, Recall 97.03%.
- **Ad dismissal** success rate: **85.96%**; **pop-up handling** success rate: **95.56%**.
- **Robustness**: detected and handled **309 UI changes**.

Evaluation — RQ3: Perf. on UNKNOWN Dataset

Table VII: ChameleoScan's detection on **UNKNOWN**



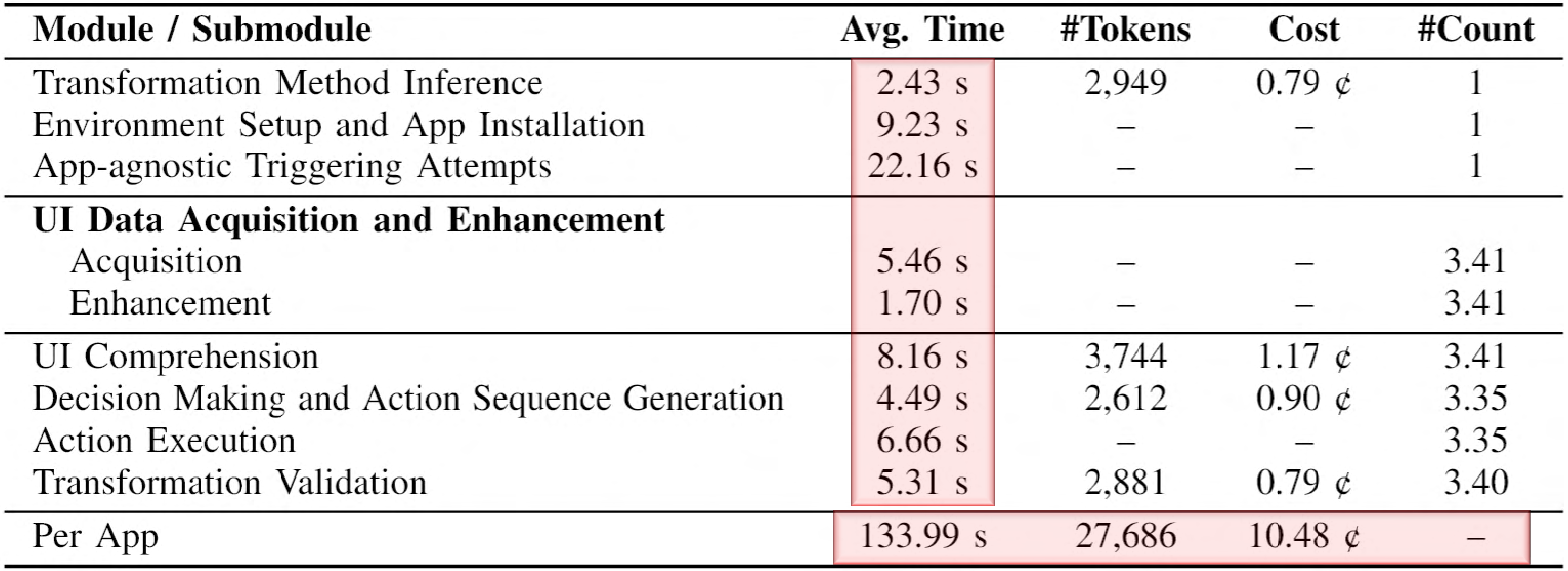
Category	#Apps (Correct)	AR	Avg. Time (s)	P (%)
Chameleon App	162 (150)	0.89	81.33	92.59
Inconclusive	1,482 (1,447)	2.13	139.74	97.53
Total	**1,644 (1,597)**	**2.01**	**133.99**	**97.14**

Insights from Newly Discovered Chameleon Apps:

- While known Chameleon apps facilitate **gambling**, **pornography**, and **content piracy**, newly identified variants also promote **game account trading platforms** and **financial services**.
- **128** newly discovered apps exhibit **similar behavioral patterns**: **automatic redirection** to external websites via **Safari** upon launch. Developer naming conventions frequently disclosed actual functionality (e.g., *"imtoken wallet app bitpie tpwallet tokenpocket TP IMT."*).

Evaluation — RQ3: Perf. on UNKNOWN Dataset

Table VI: Resource and Cost Analysis of ChameleoScan on **UNKNOWN** dataset (1,644 apps).



Module / Submodule	**Avg. Time**	**#Tokens**	**Cost**	**#Count**
Transformation Method Inference	2.43 s	2,949	0.79 ¢	1
Environment Setup and App Installation	9.23 s	–	–	1
App-agnostic Triggering Attempts	22.16 s	–	–	1
UI Data Acquisition and Enhancement				
Acquisition	5.46 s	–	–	3.41
Enhancement	1.70 s	–	–	3.41
UI Comprehension	8.16 s	3,744	1.17 ¢	3.41
Decision Making and Action Sequence Generation	4.49 s	2,612	0.90 ¢	3.35
Action Execution	6.66 s	–	–	3.35
Transformation Validation	5.31 s	2,881	0.79 ¢	3.40
Per App	133.99 s	27,686	10.48 ¢	–

Evaluation — RQ4: Comparison with Chameleon-Hunter

Chameleon-Hunter

KNOWN Dataset:

$\left.\begin{array}{l}\textbf{39.91\% (89 apps)}\text{ employ }\textbf{Flutter} \\ \textbf{27.35\% (61 apps)}\text{ employ }\textbf{WebView}\end{array}\right\}$ **59.19% (132 apps) out of courage**

UNKNOWN Dataset: **128 newly discovered apps**

(automatic redirection to external websites via Safari upon launch)

- Show a **benign** static UI
  ↓
- **Destroy** UI in **callback** handlers
  ↓
- **Dynamically** fetch **external URLs** via **CloudKit**

→ **Fully evades static analysis**

Evaluation — RQ4: Comparison with Mask-Catcher

Mask-Catcher

- **128 Chameleon apps without user reviews** (in UNKNOWN dataset) → None **flagged as suspicious**

- **Code-similarity analysis:** for **94.83%** of suspicious apps, the **maximum similarity** to any known Chameleon app is **< 0.5**, which is **comparable to the benign control group**.

→ Our collected dataset is **more complex**: these chameleon apps are **NOT simple repackaging** of **known code templates**.

Conclusion — **Takeaways**

- **We conducted systematic analysis on Chameleon apps.**
  - ✓ We constructed **a large dataset of iOS Chameleon apps** and established a **taxonomy** of **10 transformation method categories.**
- **We developed ChameleoScan from scratch.**
  - ✓ It is the **first LLM-powered automated UI exploration** system for detecting iOS Chameleon apps through synergistic integration of **App Store metadata** and **runtime UI evidence**.
  - ✓ It uses **multimodal few-shot reasoning** to handle **ads/pop-ups** and **ambiguous tasks** near **human-level**, where deterministic exploration break.
- **We evaluated ChameleoScan on both a verified benchmark and the App Store.**
  - ✓ It **discovered 150 previously unseen** Chameleon apps on **the App Store.**
  - ✓ It **outperforms** existing SOTA methods particularly in apps employing **hybrid frameworks**.
- **Chameleon apps are diverse and evolving.**
  - ✓ **We aim to contribute to a cleaner, safer mobile app ecosystem.**

End — **Thank You!**

Thank you for you attention!

Q&A

Our framework is open-sourced at https://github.com/ChameleoScan