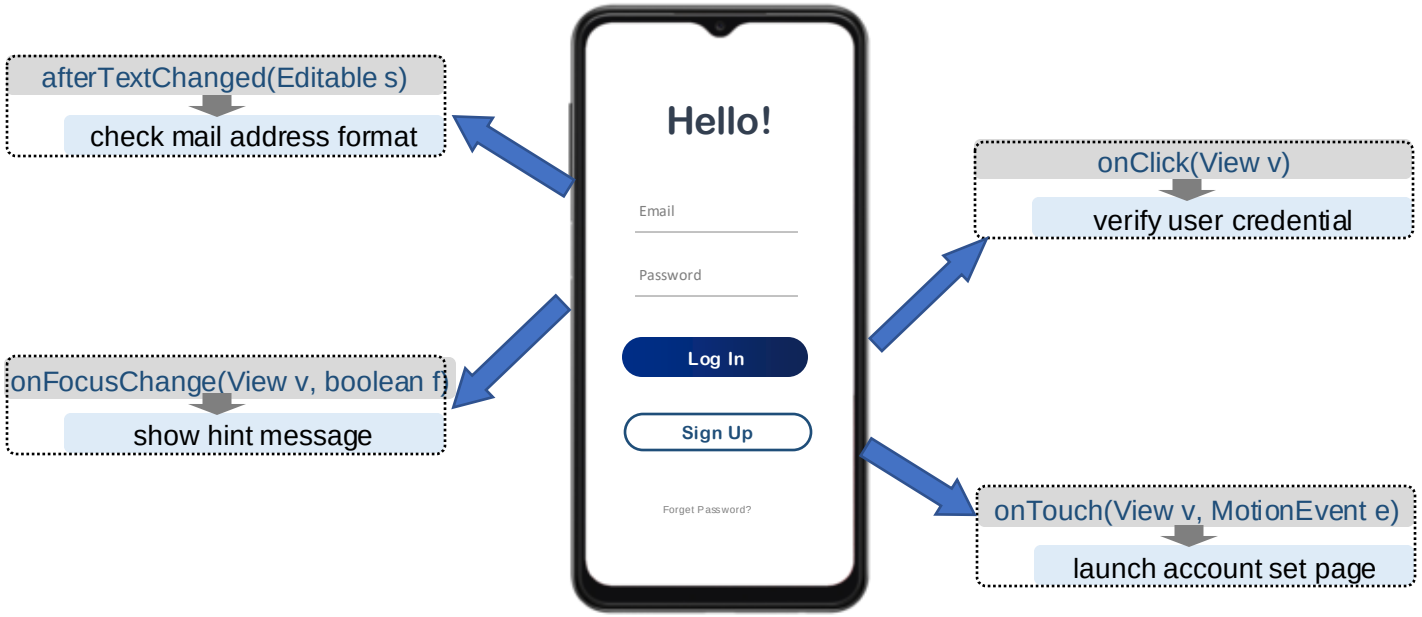
UI-CTX: Understanding UI Behaviors with Code Contexts for Mobile Applications

Jiawei Li, Jiahao Liu*, Jian Mao, Jun Zeng, Zhenkai Liang*

NDSS, February 2025

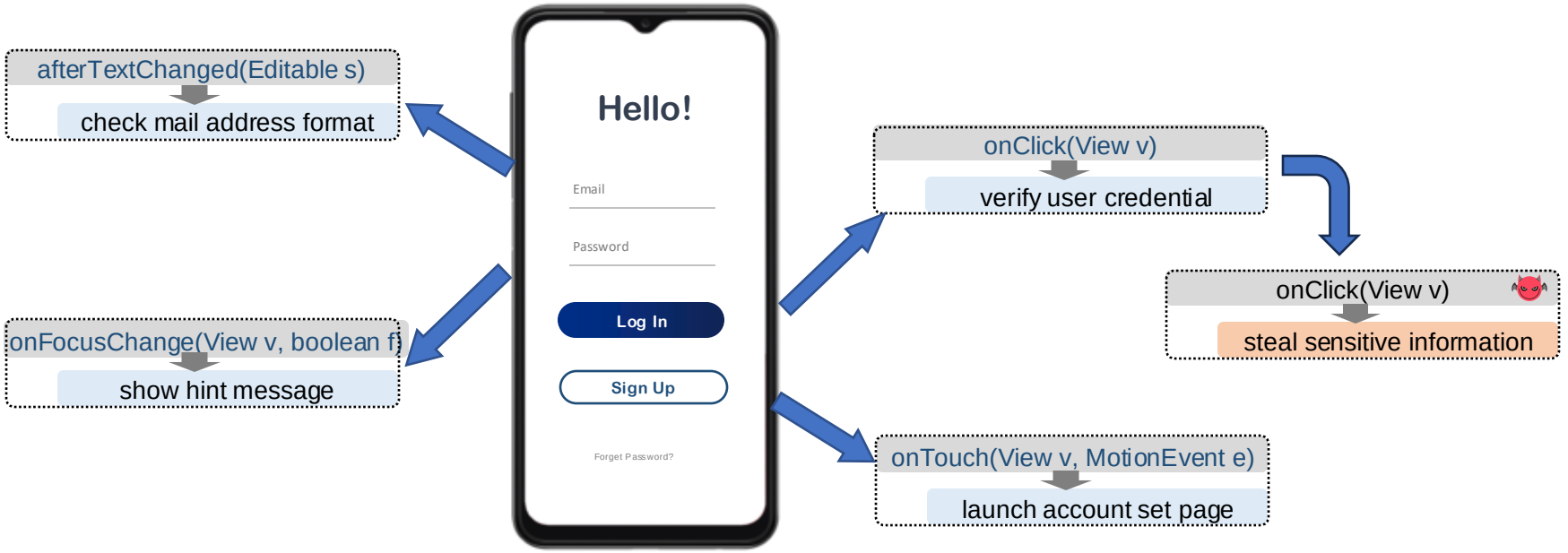
UI Behaviors in Mobile Applications

Mobile apps utilize UI widgets to interact with users and trigger specific operational functionalities



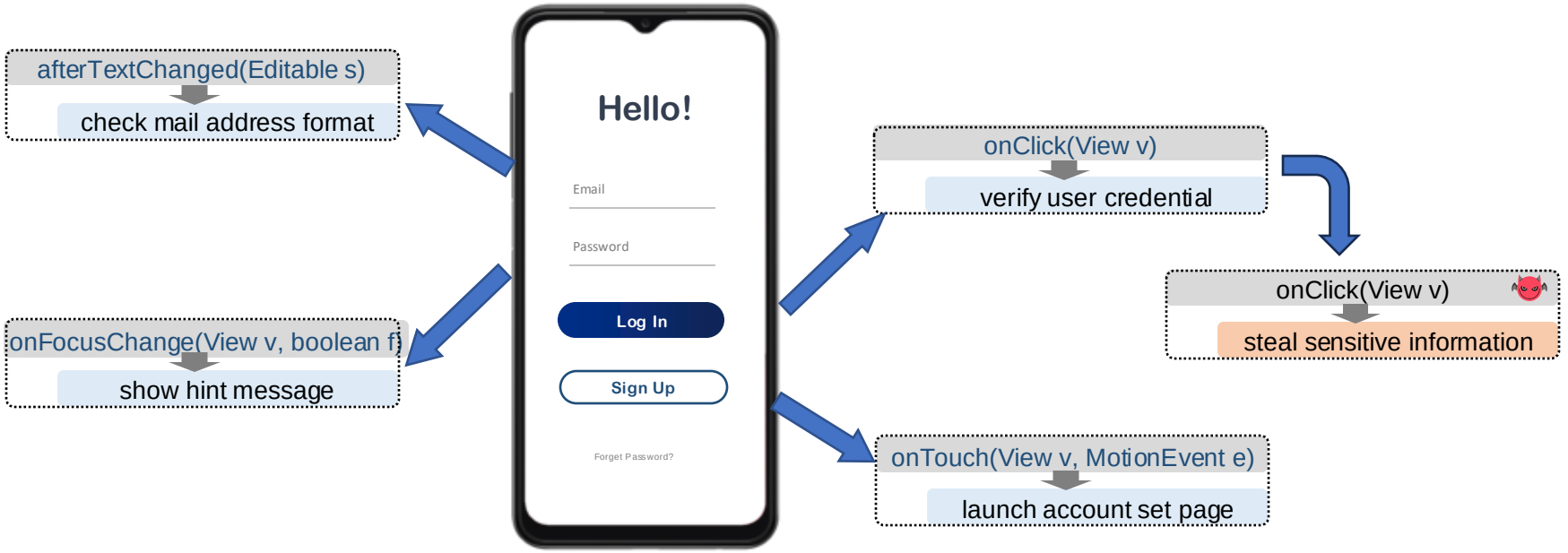
UI Behaviors in Mobile Applications

Mobile apps utilize UI widgets to interact with users and trigger specific operational functionalities



UI Behaviors in Mobile Applications

Mobile apps utilize UI widgets to interact with users and trigger specific operational functionalities



It is important to identify and understand the *intended behaviors* behind UI widgets to mitigate such potential threats

Representing UI Widgets Behaviors

Representing UI widgets' behaviors generally fall into three categories: appearance-based, permission-based, and code-based methods

- **Appearance-based**
  - Image or text on UI widgets
  - fail to recognize the real intended behaviors

Representing UI Widgets Behaviors

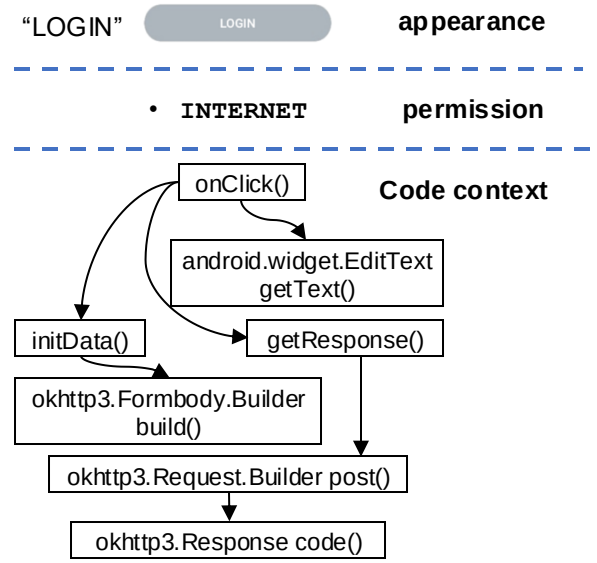
Representing UI widgets' behaviors generally fall into three categories: appearance-based, permission-based, and code-based methods

- **Appearance-based**
  - Image or text on UI widgets
  - fail to recognize the real intended behaviors
- **Permission-based**
  - Required permissions (*e.g.,* INTERNET/SMS)
  - False positives when complex behaviors share similar permissions

Representing UI Widgets Behaviors

Representing UI widgets' behaviors generally fall into three categories: appearance-based, permission-based, and code-based methods

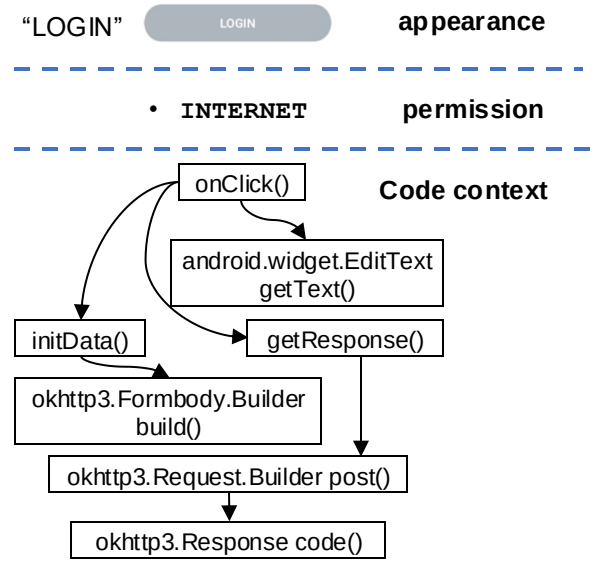
- **Appearance-based**
  - Image or text on UI widgets
  - fail to recognize the real intended behaviors
- **Permission-based**
  - Required permissions (*e.g.,* INTERNET/SMS)
  - False positives when complex behaviors share similar permissions
- **Code-based**
  - Code contexts behind UI widgets include details describing widget behaviors



Representing UI Widgets Behaviors

Representing UI widgets' behaviors generally fall into three categories: appearance-based, permission-based, and code-based methods

- **Appearance-based**
  - Image or text on UI widgets
  - fail to recognize the real intended behaviors
- **Permission-based**
  - Required permissions (*e.g.,* INTERNET/SMS)
  - False positives when complex behaviors share similar permissions
- **Code-based**
  - Code contexts behind UI widgets include details describing widget behaviors



Code contexts contain essential information and can serve as the basis for describing UI intended behaviors

Challenges

A Over-estimated widget and code context pairs

Analyze variable reachability to link widgets with code (*e.g.,* $r6 flows to all events)

```
1  // app: com.mediaaz.bryanadamtopfree
2  $r6 = invoke $r1.<findViewById(int)>(2131230766)
3  $r11 = (ImageView) $r6
4  $r6 = invoke $r1.<findViewById(int)>(2131230767)
5  $r12 = (ImageView) $r6
6  $r6 = invoke $r1.<findViewById(int)>(2131230768)
7  $r13 = (ImageView) $r6
8  ...
9  $r25 = new Gallery$Adapter$7
10 invoke $r25.<Gallery$Adapter$7: void <init>(...)
11 invoke $r11.<void setOnClickListener(...)>($r25)

12 $r26 = new Gallery$Adapter$8
13 invoke $r26.<Gallery$Adapter$8: void <init>(...)
14 invoke $r12.<void setOnClickListener(...)>($r26)

15 $r27 = new Gallery$Adapter$9
16 invoke $r27.<Gallery$Adapter$9: void <init>(...)
17 invoke $r13.<void setOnClickListener(...)>($r27)
```

Challenges

A Over-estimated widget and code context pairs

Analyze variable reachability to link widgets with code (*e.g.,* $r6 flows to all events)

Challenges

A Over-estimated widget and code context pairs

Analyze variable reachability to link widgets with code (*e.g.,* $r6 flows to all events)

```
1  // app: com.mediaaz.bryanadamtopfree
2  $r6 = invoke $r1.<findViewById(int)>(2131230766)
3  $r11 = (ImageView) $r6
4  $r6 = invoke $r1.<findViewById(int)>(2131230767)
5  $r12 = (ImageView) $r6
6  $r6 = invoke $r1.<findViewById(int)>(2131230768)
7  $r13 = (ImageView) $r6
8  ...
9  $r25 = new Gallery$Adapter$7
10 invoke $r25.<Gallery$Adapter$7: void <init>(...)
11 invoke $r11.<void setOnClickListener(...)>($r25)

12 $r26 = new Gallery$Adapter$8
13 invoke $r26.<Gallery$Adapter$8: void <init>(...)
14 invoke $r12.<void setOnClickListener(...)>($r26)

15 $r27 = new Gallery$Adapter$9
16 invoke $r27.<Gallery$Adapter$9: void <init>(...)
17 invoke $r13.<void setOnClickListener(...)>($r27)
```

Not all code in the "right" event callback can be triggered at app runtime

```
1  // app: com.bionicpanda.aquapets
2  r0 := @this: TankWallActivity
3  $r1 := @parameter0: View
4  $i0 = invoke $r1.<View: int getId()>()
5  lookupswitch($i0) {
6    case 2131427367: goto s1;
7    case 2131427707: goto s2;
8    case 2131427708: goto s3;
9    case 2131427713: goto s4;
10   default: goto return;
11 }
12 return
13 s2: invoke r0.<TankWallActivity: void finish()>()
14 return
15 s1: invoke <game.b: void a(int)>(0)
16 invoke r0.<TankWallActivity: void a(int)>(99)
17 return
18 s3: $r2 = new Intent
19 $r3 = invoke r0.<TankWallActivity: Context
       getApplicationContext()>()
20 ...
21 return
22 s4: $r4 = r0.<TankWallActivity: EditText e>
23 $r5 = invoke $r4.<EditText: Editable getText()>()
```

Challenges

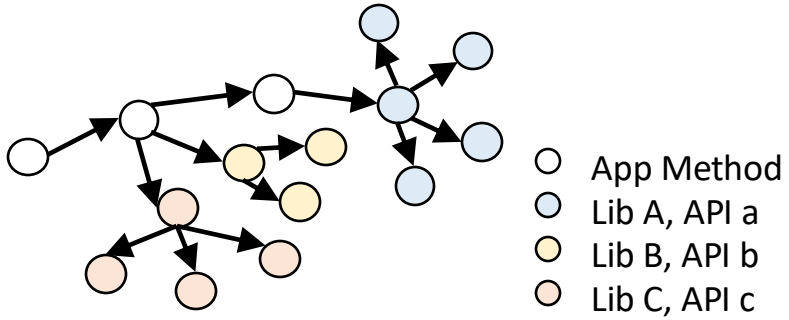
B Extensive unnecessary code contexts

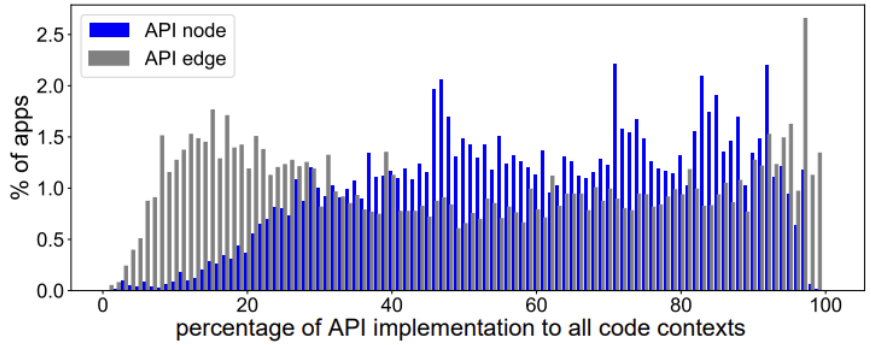
Apps often rely on third-party library (TPL) and software development kit (SDK) APIs (*e.g.,* HTTP requests) to implement their functionalities

Challenges

B Extensive unnecessary code contexts

Apps often rely on third-party library (TPL) and software development kit (SDK) APIs (*e.g.*, HTTP requests) to implement their functionalities



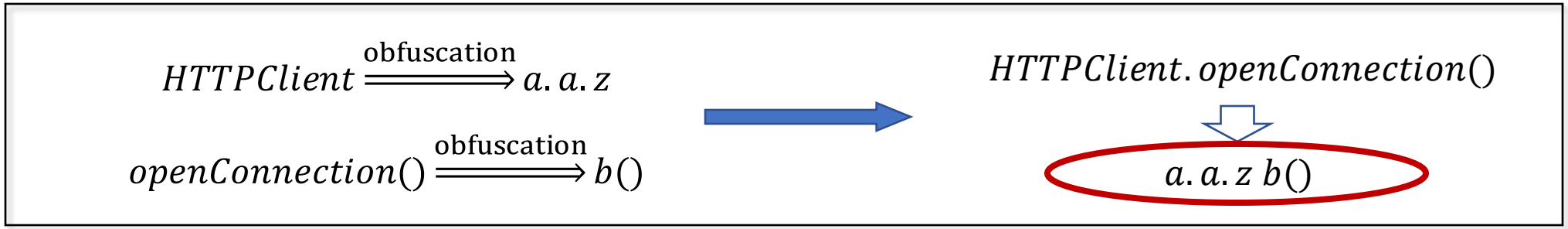


- In **88% of apps**, **>30%** of function calls originate from external TPL/SDK APIs
- In **over half of apps**, external APIs make up **>60% of the app code** *Based on 20,000 apps

Challenges

C Limited function call semantics

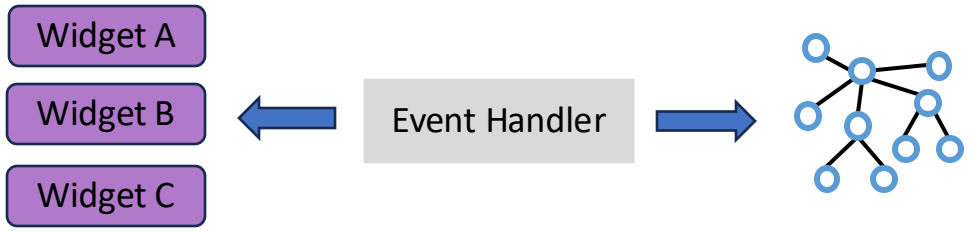
- Class and function names are commonly used to represent a function's semantics since they reasonably reflect its purpose
- Code shrinking and protection techniques obfuscate name semantics



Limited meaningful code semantics

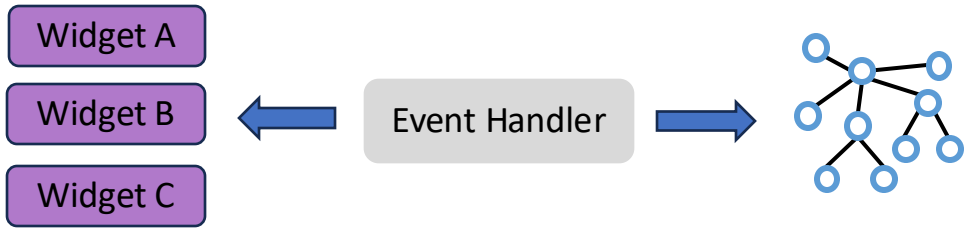
Our Insights

- Begin with the event handler to explore widget-code pairs



Our Insights

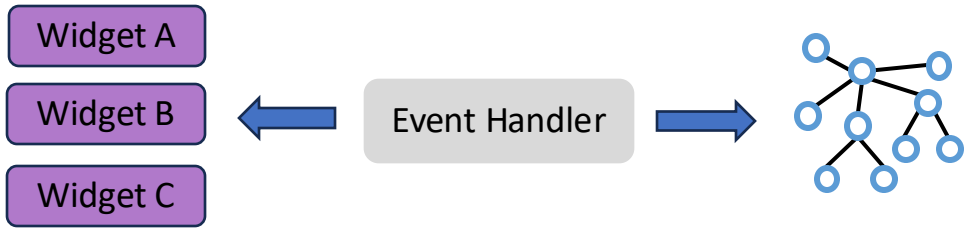
- Begin with the event handler to explore widget-code pairs



- Abstract and summarize external libraries to reduce unnecessary code, allowing a focused analysis of widget behaviors

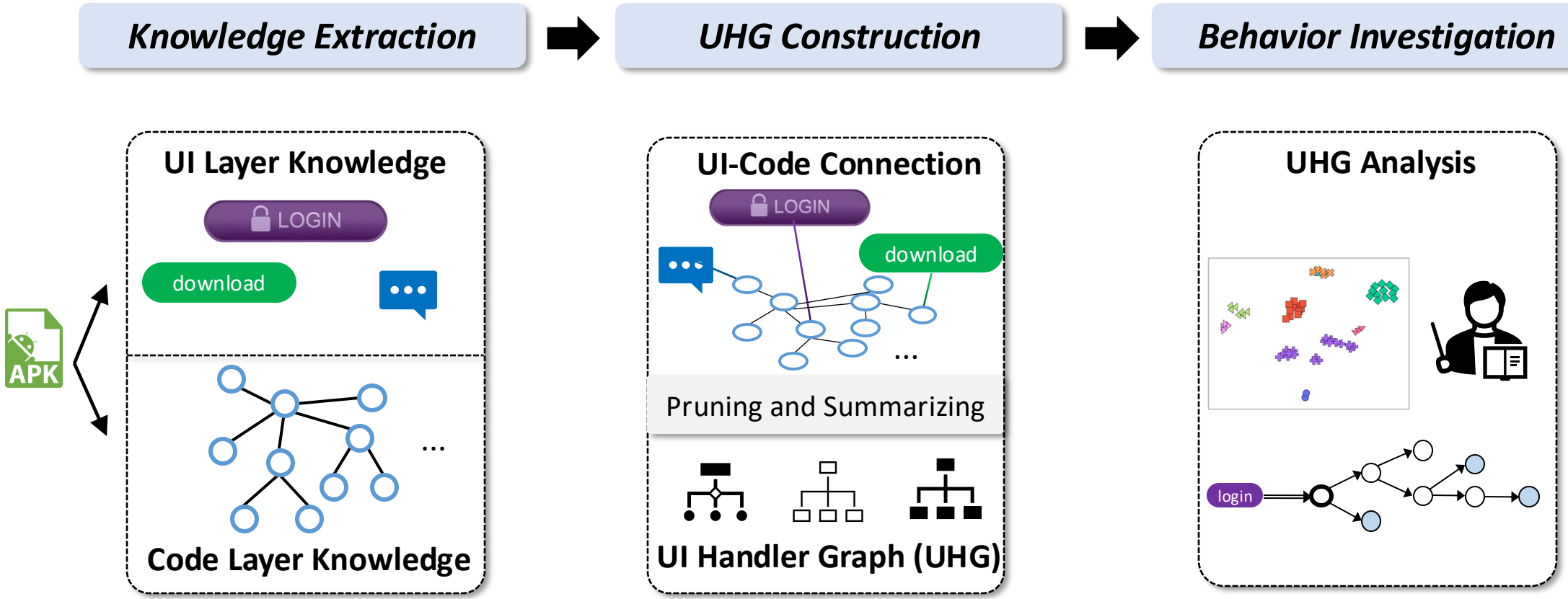
Our Insights

- Begin with the event handler to explore widget-code pairs



- Abstract and summarize external libraries to reduce unnecessary code, allowing a focused analysis of widget behaviors
- Instruction opcodes serve as the fundamental units of functions, providing stability and robustness

UI-CTX Overview



Knowledge Extraction

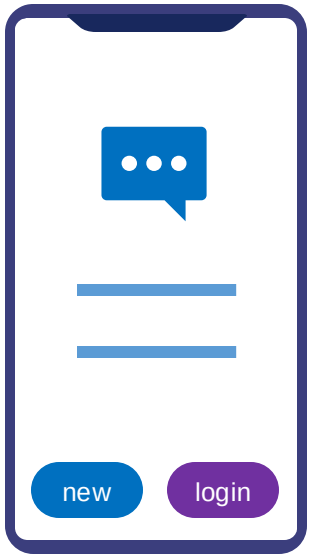
Leverage both UI and code layer knowledge to build UHG

Knowledge Extraction

Leverage both UI and code layer knowledge to build UHG

UI Layer Semantics

Layout Id	Widget Id	Text	Image



Layout id
0x7f0500c5

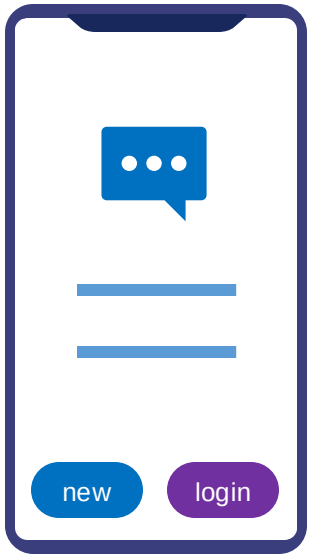
layout/activity_login
ImageView
src: "Logo"
id: 2131493103
Button
id: 2131493104
Button
text: "login"
id: 2131493105
...

Knowledge Extraction

Leverage both UI and code layer knowledge to build UHG

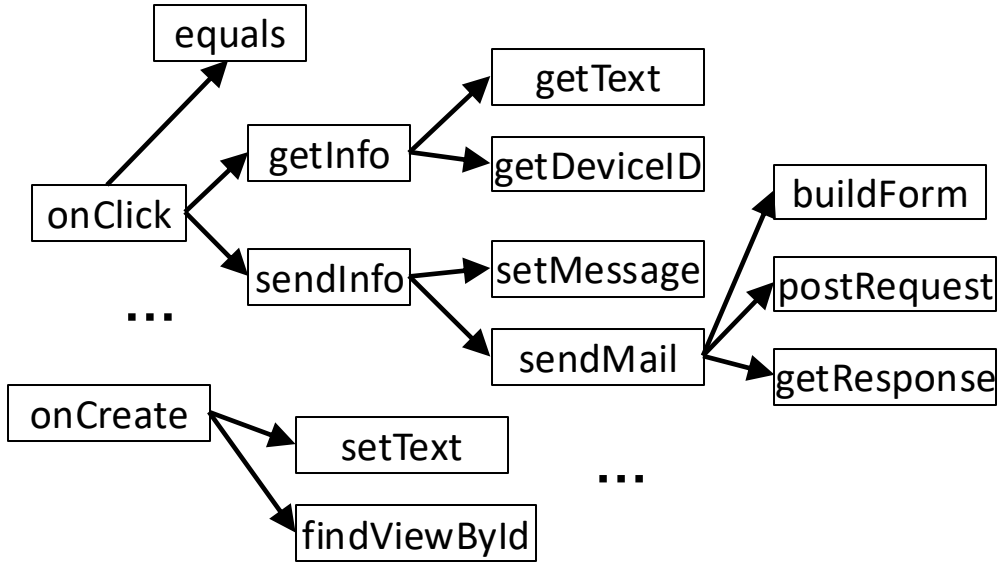
UI Layer Semantics			
Layout Id	Widget Id	Text	Image

Code Layer Semantics		
Event	Method Implement	Code Block



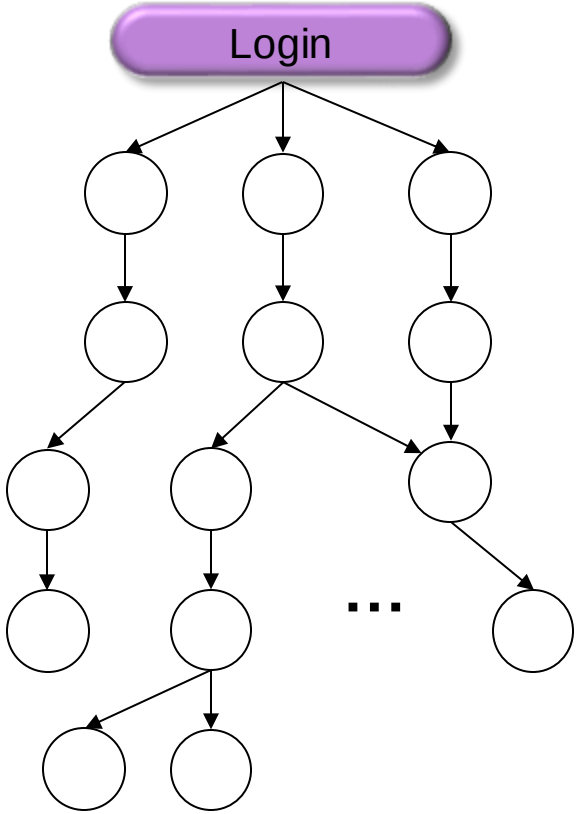
Layout id
0x7f0500c5

layout/activity_login
ImageView
src: "Logo"
id: 2131493103
Button
id: 2131493104
Button
text: "login"
id: 2131493105
...



UHG Construction

Make the representation precise and concise

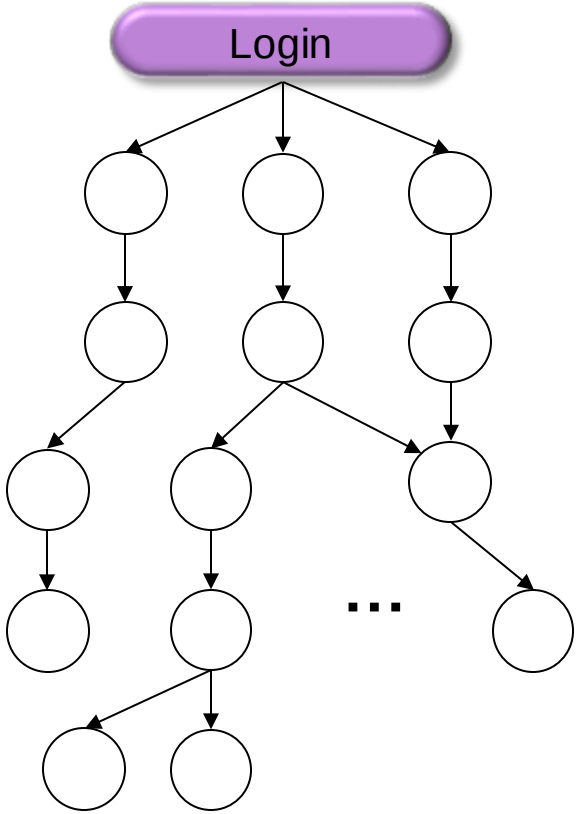



```
public class MainActivity extends Activity {
  private android.widget.Button btnLogin;
  private static final String BTN_TEXT;
  public void onCreate() {
    MainActivity r0;
    r0 := <MainActivity: String BTN_TEXT>
    invoke r0.<Activity: void setContentView(int)>(ID_X);
    r0 := @this: MainActivity;
    $r1 = invoke r0.<Activity: View findViewById(int)>(ID_Y);
    $r2 = (android.widget.Button) $r1
    r0.<MainActivity: Button btnLogin> = $r2;
    View$OnClickListener $r1;
    r0 := @this: MainActivity;
    $r1 = new MainActivity$1;
    invoke $r1.<MainActivity$1: void <init>(...);
    $r2 = r0.<MainActivity: Button btnLogin>;
    invoke $r2.<void setOnClickListener...>($r1);
    return;
  } ... }
```

Event Set

UHG Construction

Make the representation precise and concise



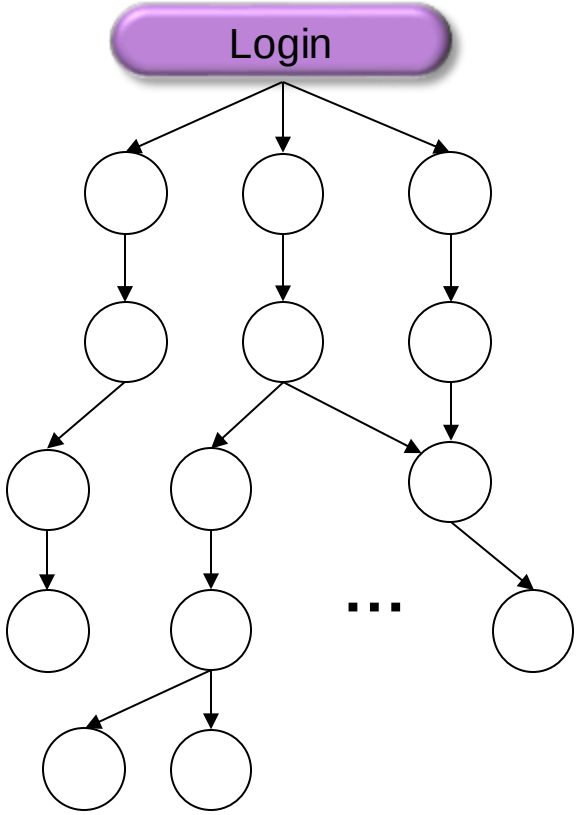
Backward Tracking

Event Set

```
public class MainActivity extends Activity {
  private android.widget.Button btnLogin;
  private static final String BTN_TEXT;
  public void onCreate() {
    MainActivity r0;
    r0 := <MainActivity: String BTN_TEXT>
    invoke r0.<Activity: void setContentView(int)>(ID_X);
    r0 := @this: MainActivity;
    $r1 = invoke r0.<Activity: View findViewById(int)>(ID_Y);
    $r2 = (android.widget.Button) $r1
    r0.<MainActivity: Button btnLogin> = $r2;
    View$OnClickListener $r1;
    r0 := @this: MainActivity;        ⟨——, MainActivity$1.onClick()⟩
    $r1 = new MainActivity$1;
    invoke $r1.<MainActivity$1: void <init>(...);
    $r2 = r0.<MainActivity: Button btnLogin>;
    invoke $r2.<void setOnClickListener...>($r1);
    return;
  } ... }
```

UHG Construction

Make the representation precise and concise



Backward Tracking

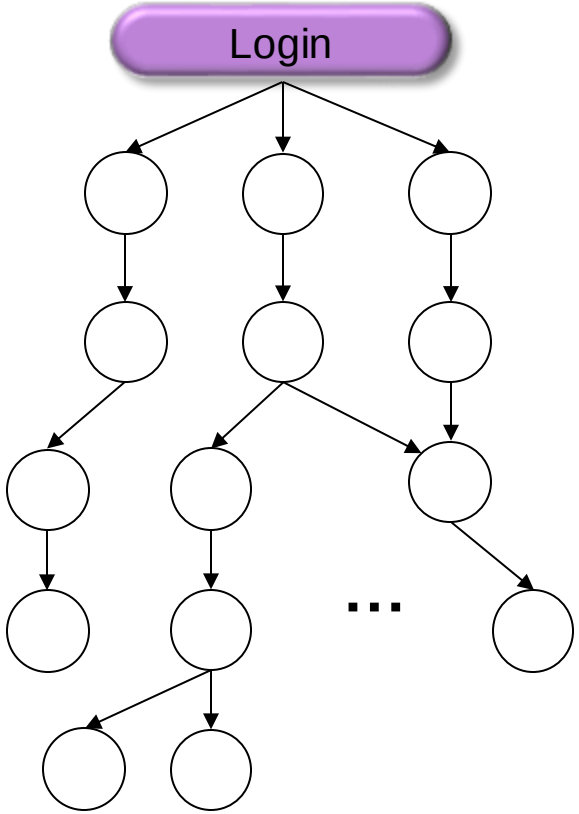
Event Set

```
public class MainActivity extends Activity {
  private android.widget.Button btnLogin;
  private static final String BTN_TEXT;
  public void onCreate() {
    MainActivity r0;
    r0 := <MainActivity: String BTN_TEXT>
    invoke r0.<Activity: void setContentView(int)>(ID_X);
    r0 := @this: MainActivity;
    $r1 = invoke r0.<Activity: View findViewById(int)>(ID_Y);
    $r2 = (android.widget.Button) $r1
    r0.<MainActivity: Button btnLogin> = $r2;
    View$OnClickListener $r1;
    r0 := @this: MainActivity;
    $r1 = new MainActivity$1;
    invoke $r1.<MainActivity$1: void <init>(...);
    $r2 = r0.<MainActivity: Button btnLogin>;
    invoke $r2.<void setOnClickListener...>($r1);
    return;
  } ... }
```

⟨——, MainActivity$1.onClick()⟩

UHG Construction

Make the representation precise and concise



Backward Tracking

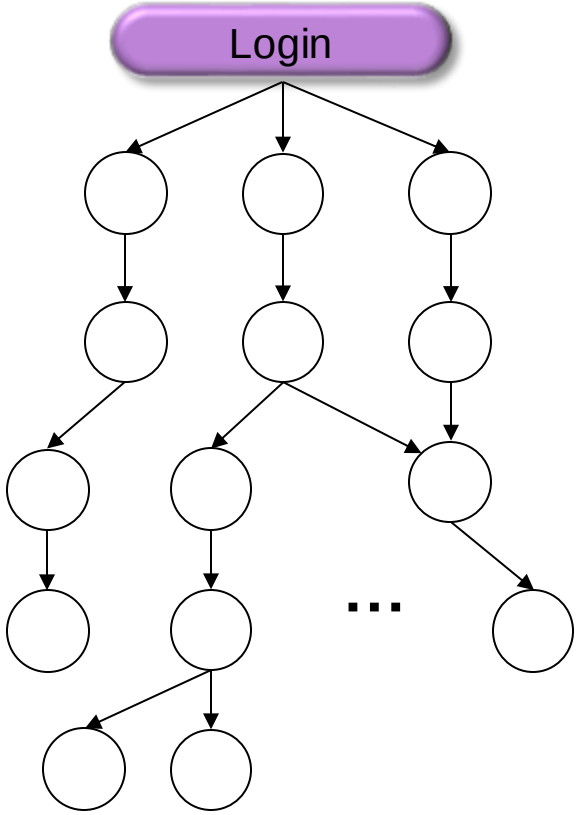
Event Set

```
public class MainActivity extends Activity {
  private android.widget.Button btnLogin;
  private static final String BTN_TEXT;
  public void onCreate() {
    MainActivity r0;
    r0 := <MainActivity: String BTN_TEXT>
    invoke r0.<Activity: void setContentView(int)>(ID_X);
    r0 := @this: MainActivity;
    $r1 = invoke r0.<Activity: View findViewById(int)>(ID_Y);
    $r2 = (android.widget.Button) $r1
    r0.<MainActivity: Button btnLogin> = $r2;
    View$OnClickListener $r1;
    r0 := @this: MainActivity;
    $r1 = new MainActivity$1;
    invoke $r1.<MainActivity$1: void <init>(...);
    $r2 = r0.<MainActivity: Button btnLogin>;
    invoke $r2.<void setOnClickListener...>($r1);
    return;
  } ... }
```

⟨——, MainActivity$1.onClick()⟩

UHG Construction

Make the representation precise and concise



Widget Id

Backward Tracking

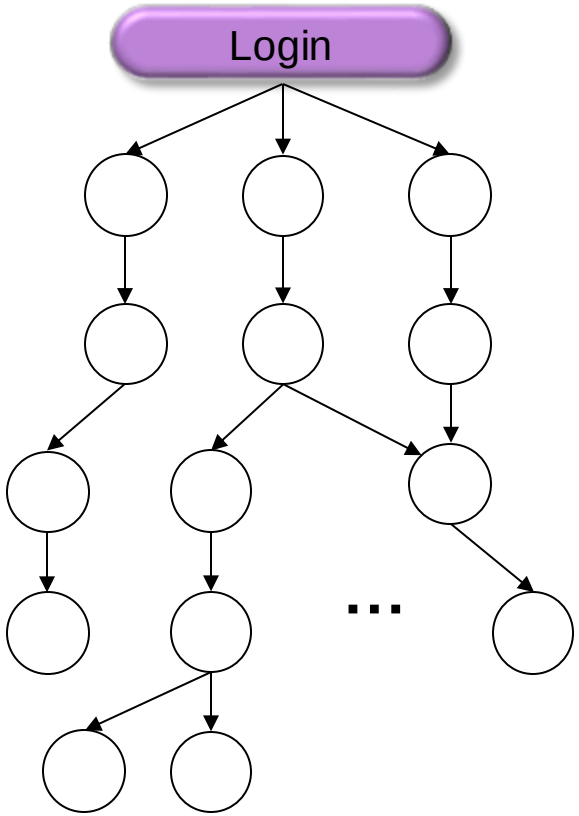
Event Set

```
public class MainActivity extends Activity {
  private android.widget.Button btnLogin;
  private static final String BTN_TEXT;
  public void onCreate() {
    MainActivity r0;
    r0 := <MainActivity: String BTN_TEXT>
    invoke r0.<Activity: void setContentView(int)>(ID_X);
    r0 := @this: MainActivity;
    $r1 = invoke r0.<Activity: View findViewById(int)>(ID_Y);
    $r2 = (android.widget.Button) $r1
    r0.<MainActivity: Button btnLogin> = $r2;
    View$OnClickListener $r1;
    r0 := @this: MainActivity;
    $r1 = new MainActivity$1;
    invoke $r1.<MainActivity$1: void <init>(...);
    $r2 = r0.<MainActivity: Button btnLogin>;
    invoke $r2.<void setOnClickListener...>($r1);
    return;
  } ... }
```

⟨ID_Y, MainActivity$1.onClick()⟩

UHG Construction

Make the representation precise and concise

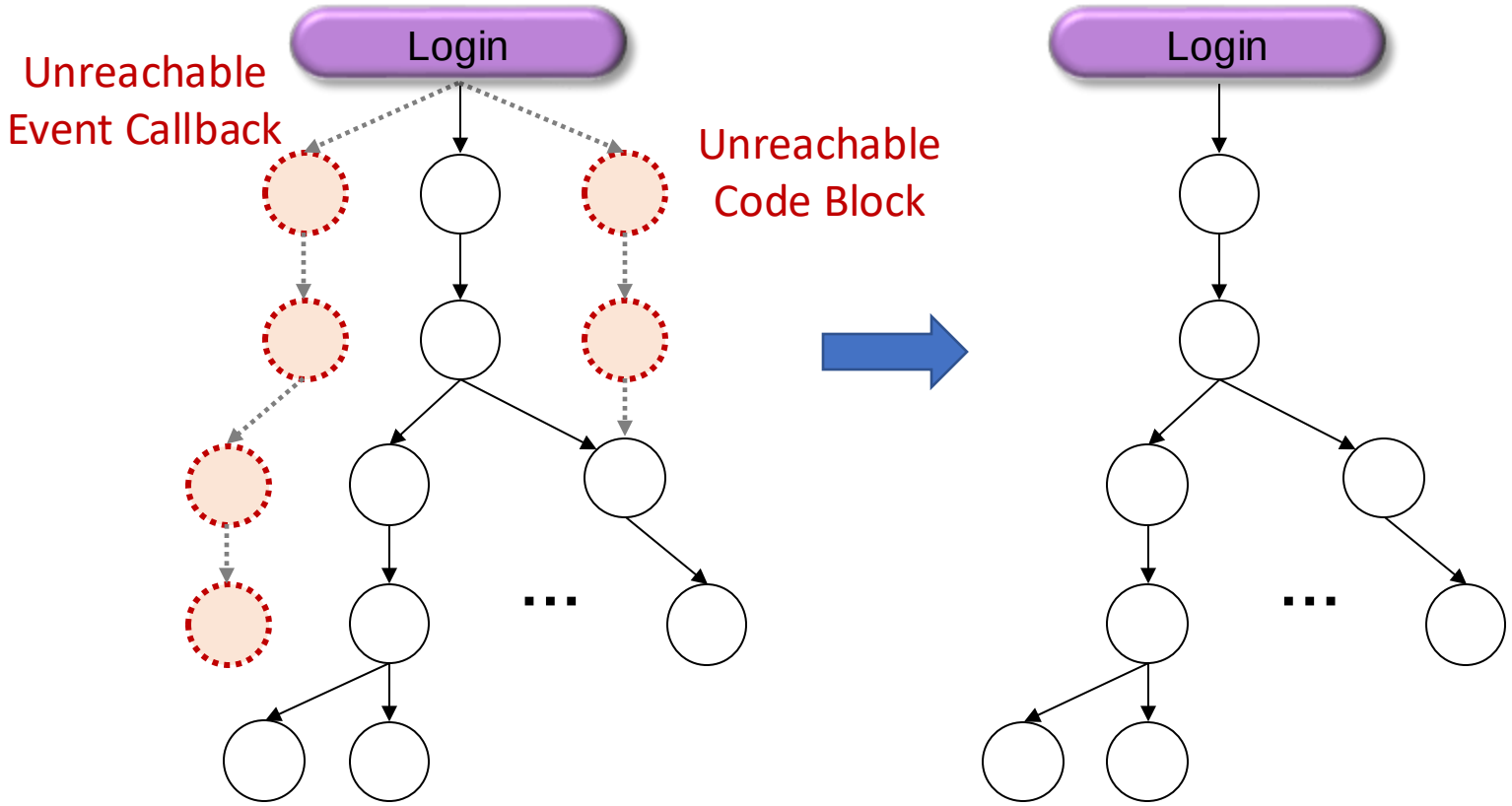



```
r0 := @this: TankWallActivity
$r1 := @parameter0: View
$i0 = invoke $r1.<View: int getId()>()
lookupswitch($i0) {
  case 2131427367: goto s1;
  case 2131427707: goto s2;
  case 2131427708: goto s3;
  case 2131427713: goto s4;
  default: goto return;
}
return
s2: invoke r0.<TankWallActivity: void finish()>()
return
s1: invoke <game.b: void a(int)>(0)
invoke r0.<TankWallActivity: void a(int)>(99)
return
s3: $r2 = new Intent
$r3 = invoke r0.<TankWallActivity: Context
getApplicationContext()>() ...
return
s4: $r4 = r0.<TankWallActivity: EditText e>
$r5 = invoke $r4.<EditText: Editable getText()>()
```

Code Block for Id: 2131427367

UHG Construction

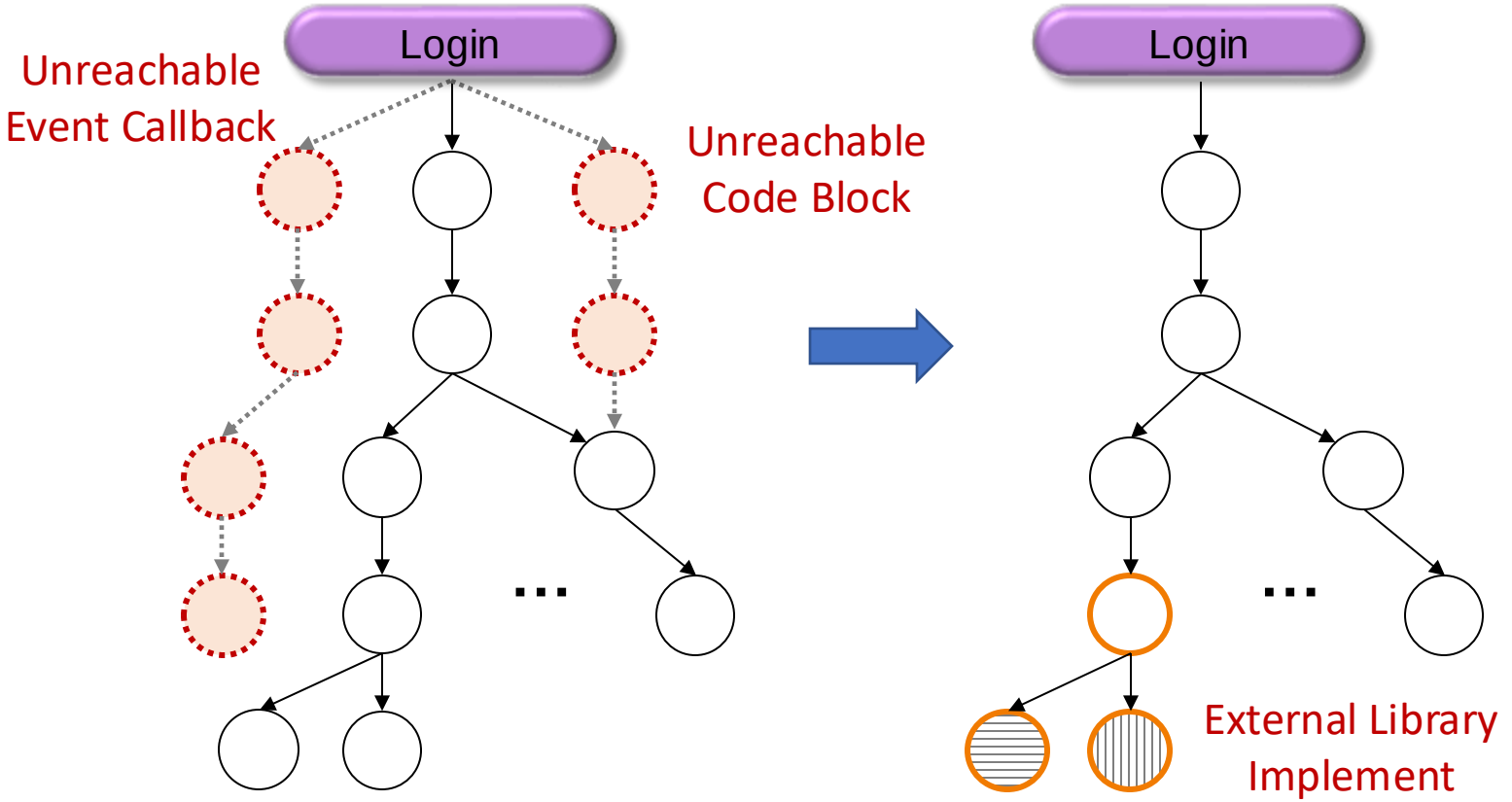
Make the representation precise and concise



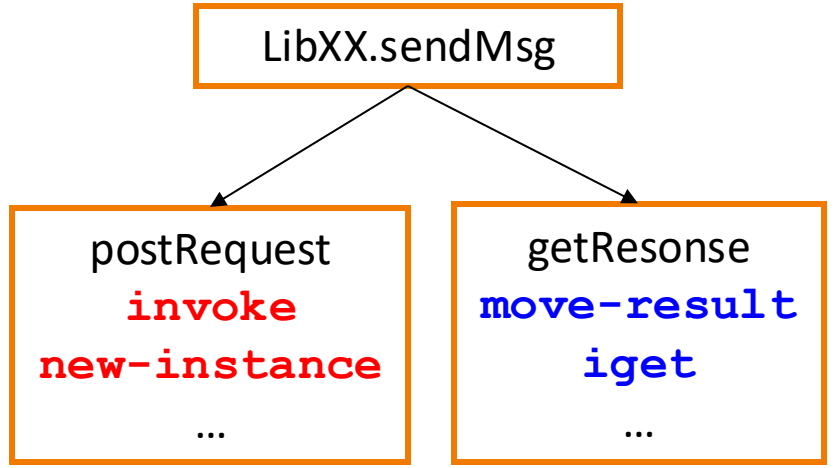
Prune unreachable code context

UHG Construction

Make the representation precise and concise

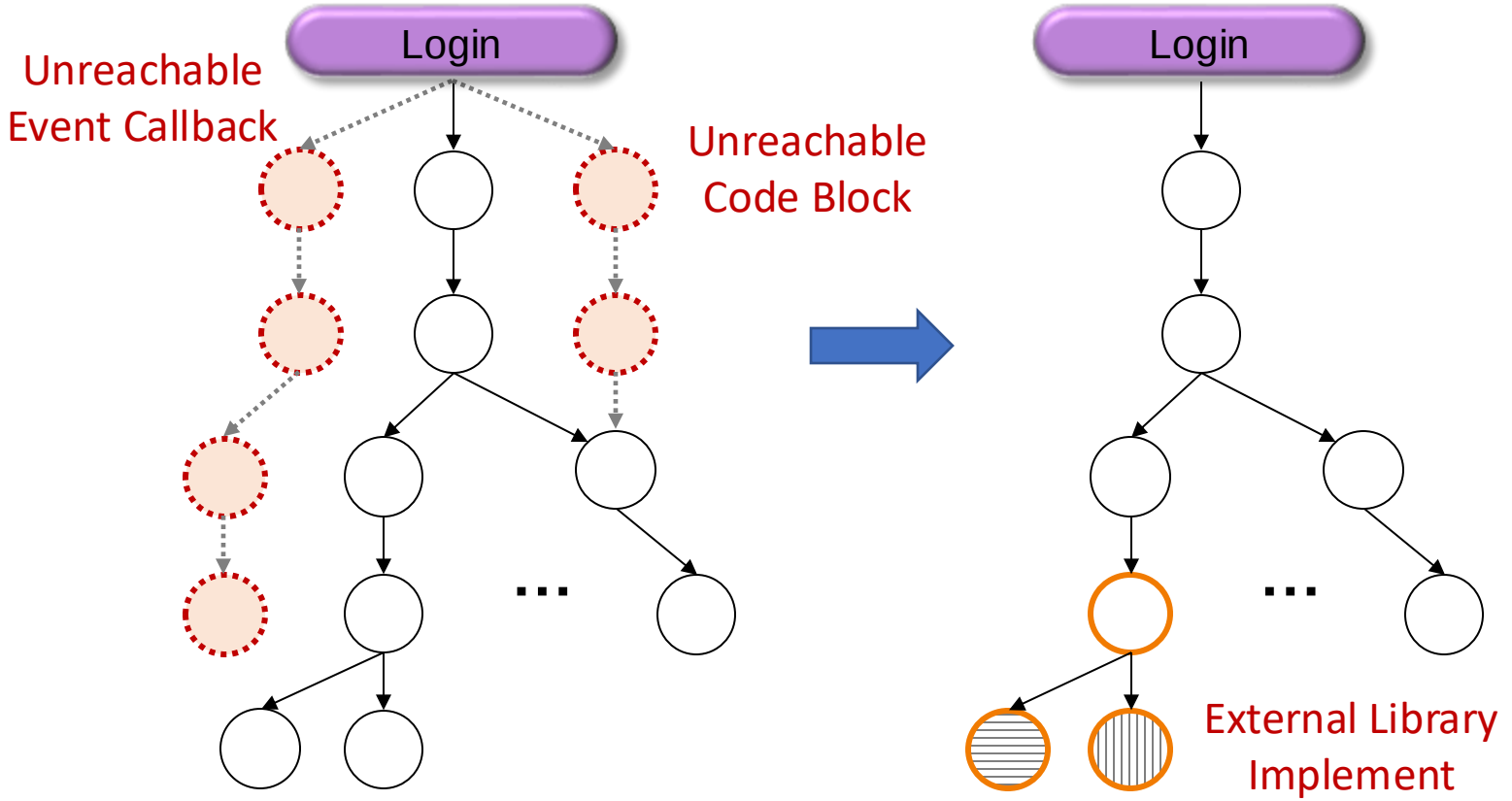


Prune unreachable code context

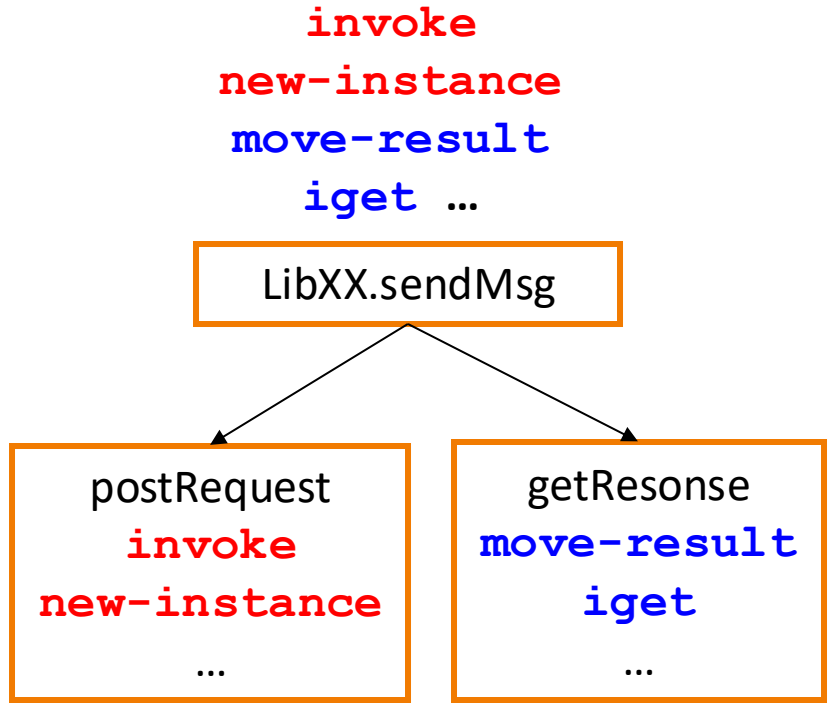


UHG Construction

Make the representation precise and concise

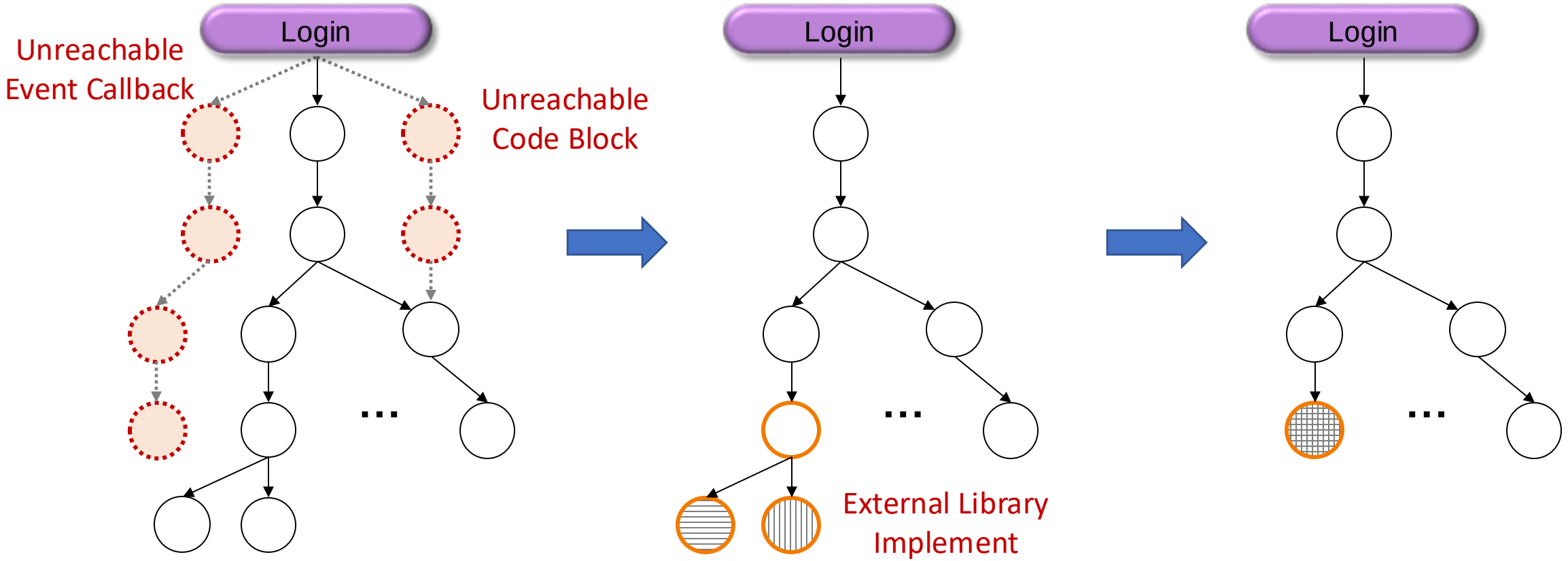


Prune unreachable code context



UHG Construction

Make the representation precise and concise



Prune unreachable code context

Summarize extensive code information

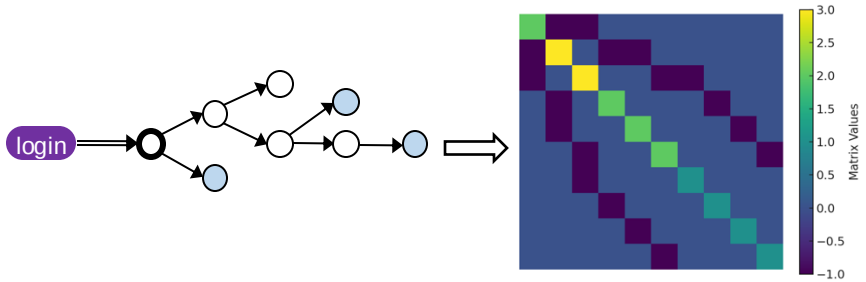
Behavior Investigation

Distill both structural and node features into a meaningful and informative representation

Behavior Investigation

Distill both structural and node features into a meaningful and informative representation

- Eigenvalues of graph Laplacian matrix capture the **global structural information** (community structure and connectivity pattern)

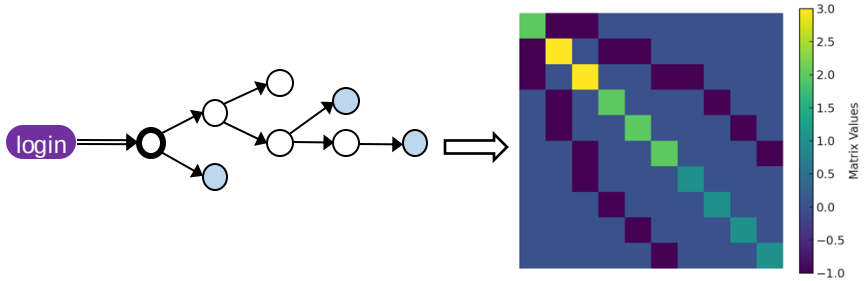


Structural information

Behavior Investigation

Distill both structural and node features into a meaningful and informative representation

- Eigenvalues of graph Laplacian matrix capture the **global structural information** (community structure and connectivity pattern)
- Opcodes capture **instruction-level** node features



Structural information

```
INVOKE_DIRECT
NEW_INSTANCE
CONST_STRING
IF_EQZ
SGET_BYTE
ADD_INT_LIT16
```

$\mu(X)$ Average pooling: **the central tendency**

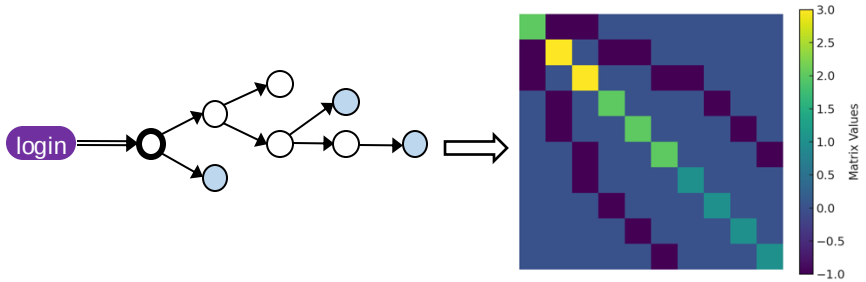
$\sigma(X)$ Standard deviation: **the dispersion**

Node information

Behavior Investigation

Distill both structural and node features into a meaningful and informative representation

- Eigenvalues of graph Laplacian matrix capture the **global structural information** (community structure and connectivity pattern)
- Opcodes capture **instruction-level** node features



Structural information

```
INVOKE_DIRECT
NEW_INSTANCE
CONST_STRING
IF_EQZ
SGET_BYTE
ADD_INT_LIT16
```

$\mu(X)$ Average pooling: **the central tendency**

$\sigma(X)$ Standard deviation: **the dispersion**

Node information

- Group UI widgets with similar functionalities based on UHG

Evaluation

Evaluation Setup

- Collect 40,000 Android apps from AndroZoo
  - Spanning 10 years (4000 per year)
  - Various market sources (Google Play, VirusShare, Mi, and Anzhi)
  - UI widgets related to account management / data operation

Evaluation

Evaluation Setup

- Collect 40,000 Android apps from AndroZoo
  - Spanning 10 years (4000 per year)
  - Various market sources (Google Play, VirusShare, Mi, and Anzhi)
  - UI widgets related to account management / data operation

Evaluation Aspects

- How UHG performs in widget behavior description compared to existing representations?
- To what extent do the different design choices in UI-CTX contribute to its performance?
- Can UI-CTX facilitate real-world app analysis?

Effectiveness of Describing UI Widget Behaviors

- Compare UI Handler Graph (UHG) with other representations (F1-score)
  - Baselines: Permission set, Call sequence

Category	delete	login	logout	send	search	download	share	save
Permission	0.29	0.36	0.37	0.60	0.31	0.47	0.81	0.38
Sequence	0.65	0.72	0.97	0.68	0.63	0.76	0.91	0.63
UHG	**0.69**	**0.78**	**0.98**	**0.72**	**0.73**	**0.86**	**0.93**	**0.71**

Effectiveness of Describing UI Widget Behaviors

- Compare UI Handler Graph (UHG) with other representations (F1-score)
  - Baselines: Permission set, Call sequence

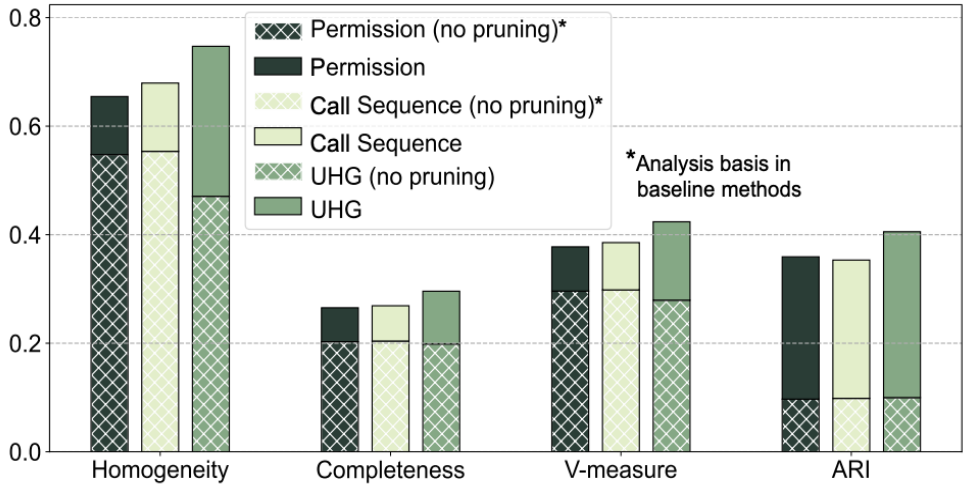
Category	delete	login	logout	send	search	download	share	save
Permission	0.29	0.36	0.37	0.60	0.31	0.47	0.81	0.38
Sequence	0.65	0.72	0.97	0.68	0.63	0.76	0.91	0.63
UHG	**0.69**	**0.78**	**0.98**	**0.72**	**0.73**	**0.86**	**0.93**	**0.71**

UI Handler Graph (UHG) is effective in describing widget behaviors

Contribution Analysis of Different Design Choices

Explore the contributions of different design choices in UI-CTX

- Code Context pruning, API Summarization, and UHG Embedding

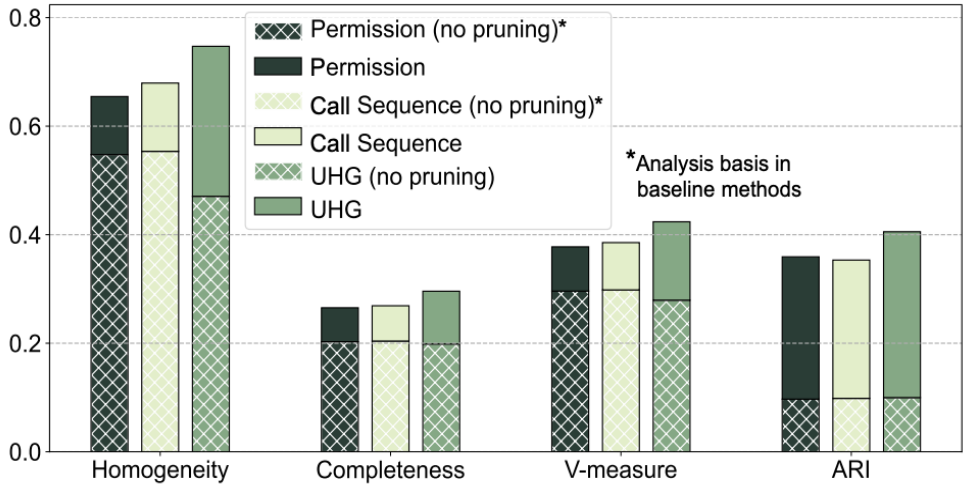


- Context pruning identifies accurate and concise code contexts

Contribution Analysis of Different Design Choices

Explore the contributions of different design choices in UI-CTX

- Code Context pruning, API Summarization, and UHG Embedding



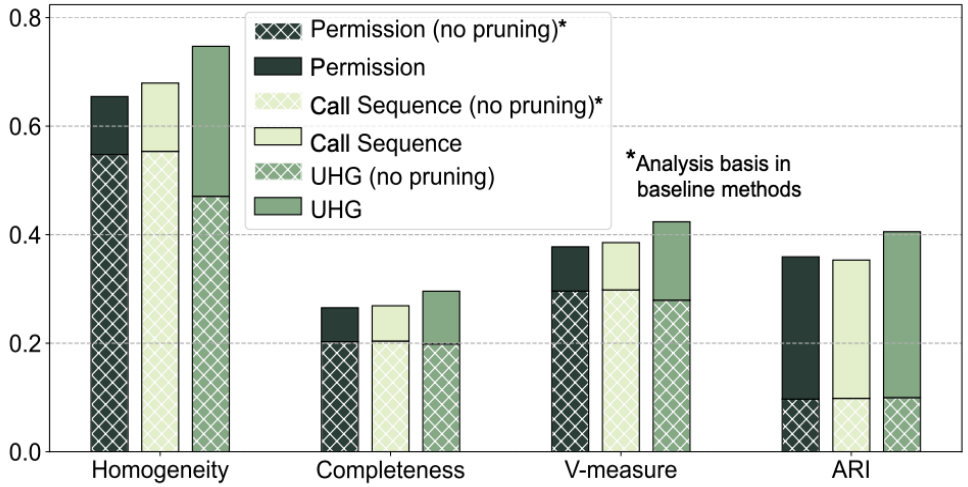
API hop	∞	6	5	4	3	2	1
% of Node	100	61.7	56.0	49.8	42.7	36.3	**32.1**
% of Edge	100	52.5	47.2	41.9	36.5	31.3	**27.9**
F1-score	0.76	0.77	0.78	0.79	0.79	0.80	**0.81**

- Context pruning identifies accurate and concise code contexts
- API Summarization reduces unnecessary external code

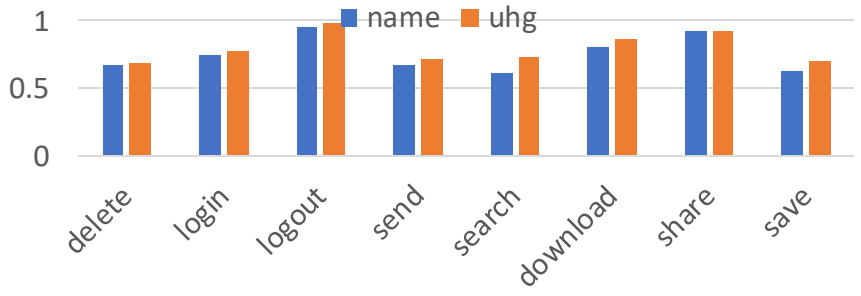
Contribution Analysis of Different Design Choices

Explore the contributions of different design choices in UI-CTX

- Code Context pruning, API Summarization, and UHG Embedding



API hop	∞	6	5	4	3	2	1
% of Node	100	61.7	56.0	49.8	42.7	36.3	**32.1**
% of Edge	100	52.5	47.2	41.9	36.5	31.3	**27.9**
F1-score	0.76	0.77	0.78	0.79	0.79	0.80	**0.81**



- Context pruning identifies accurate and concise code contexts
- API Summarization reduces unnecessary external code
- Opcodes provide more stable and informative function semantics

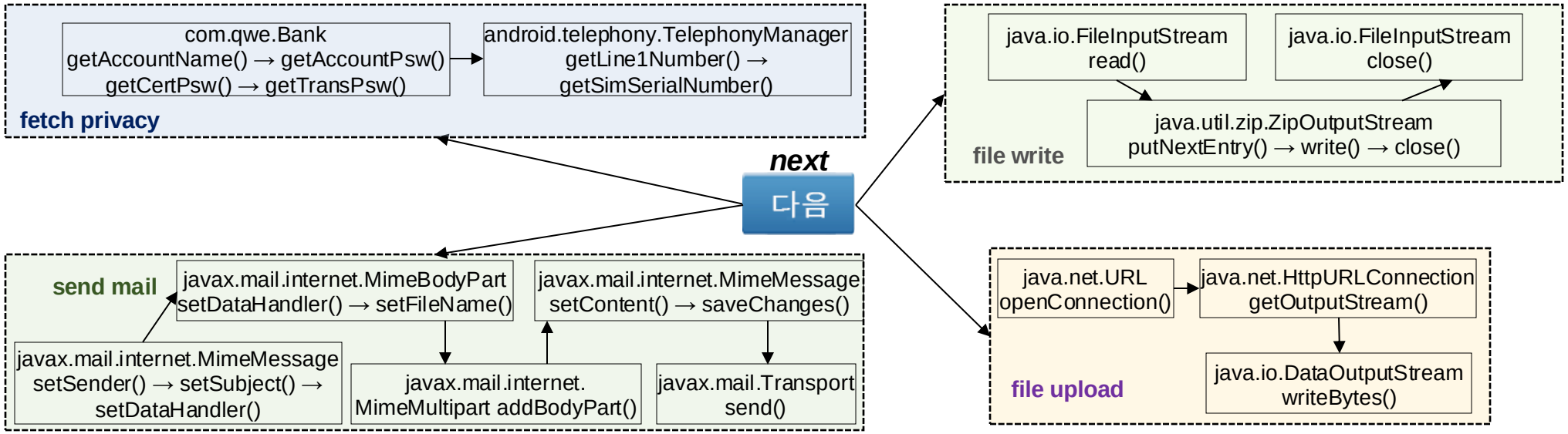
Case Study

Analyze the functionality of a next button in a real-world malicious app

Case Study

Analyze the functionality of a next button in a real-world malicious app

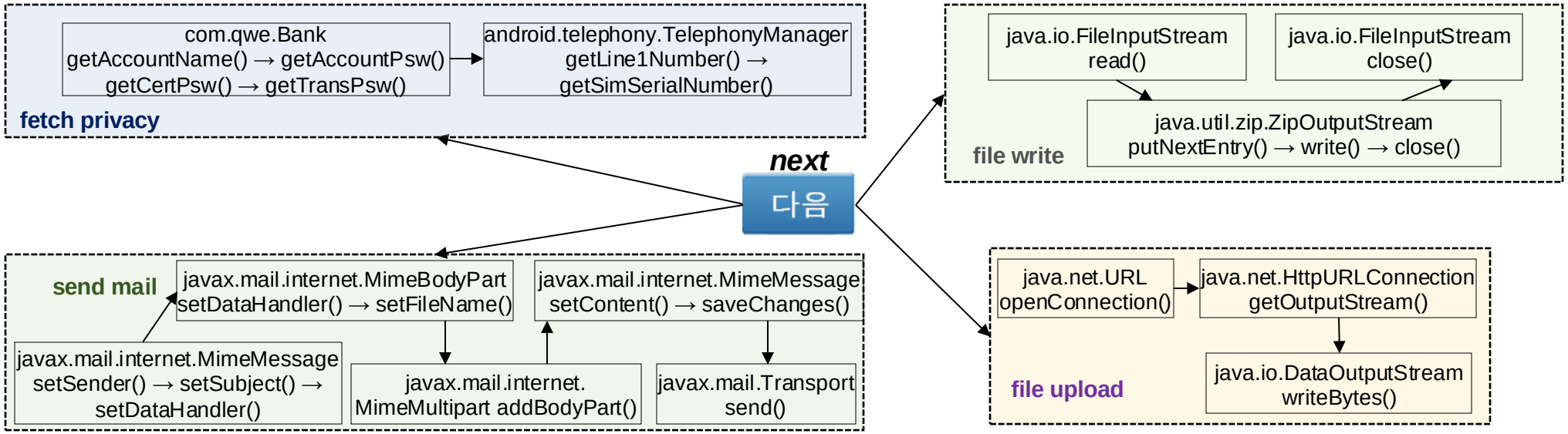
- It initiates a series of privacy stealing actions, including collecting phone number, zipping them into a file and sending them via HTTP and email.



Case Study

Analyze the functionality of a next button in a real-world malicious app

- It initiates a series of privacy stealing actions, including collecting phone number, zipping them into a file and sending them via HTTP and email.



UI-CTX can accurately present the malicious behavior behind the next button, facilitating security analysis

Summary

In this work, we:

- highlight the importance and identify the challenges of UI widget behavior representation, *e.g.,* identifying correct code contexts describing UI widget
- design UI Handler graph (UHG), a concise and accurate representation to describe UI widget behavior
- present UI-CTX, an end-to-end approach to build UHG, summarize and embed the semantics in UHG to investigate UI widget functionality

https://github.com/DaweiX/UI-CTX