Beyond the Surface:

Uncovering the Unprotected Components of Android Against Overlay Attack

Authors: Hao Zhou[1], Shuohan Wu[1], Chenxiong Qian[2], Xiapu Luo[1], Haipeng Cai[3], Chao Zhang[4]

Presenter: Song Liao[5]

[1]The Hong Kong Polytechnic University, [2]University of Hong Kong, [3]Washington State University

[4]Tsinghua University, [5]Clemson University

NDSS SYMPOSIUM/2024

Presented by Internet Society

#NDSSSymposium2024

Overlay (Floating Window)

- Overlay is one of the key UI features of Android

Overlay (Floating Window)

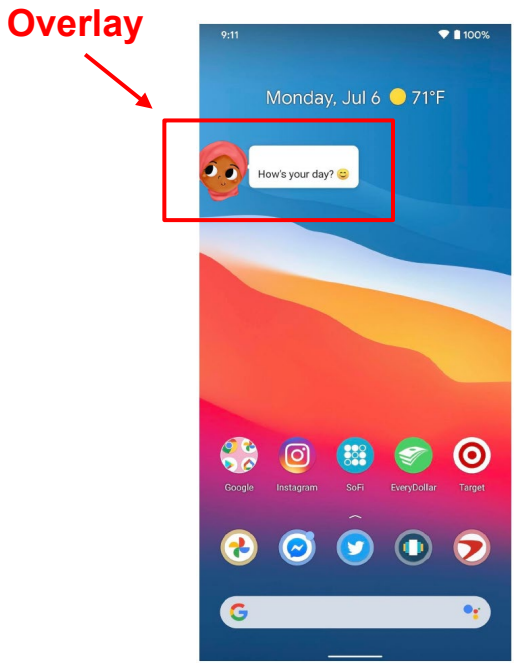
- Overlay is one of the key UI features of Android
  - Allow an app to draw over other apps' windows

Overlay (Floating Window)

- Overlay is one of the key UI features of Android
  - Allow an app to draw over other apps' windows
  - Used by many apps to enhance user experience

Overlay (Floating Window)

- Overlay is one of the key UI features of Android
  - Allow an app to draw over other apps' windows
  - Used by many apps to enhance user experience

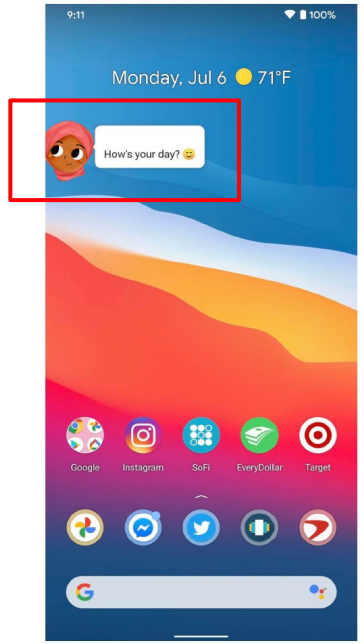


Facebook Messenger creates an overlay to let users access the received messages conveniently

Overlay (Floating Window)

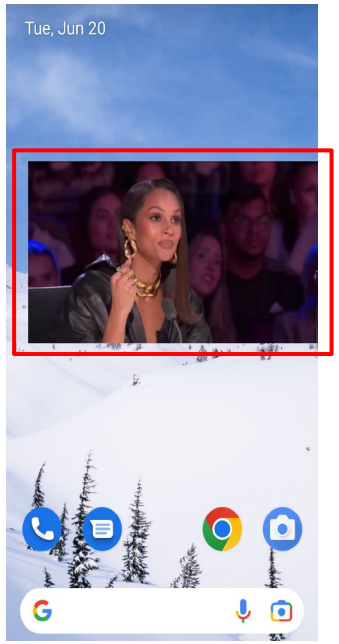
- Overlay is one of the key UI features of Android
  - Allow an app to draw over other apps' windows
  - Used by many apps to enhance user experience

Overlay



Facebook Messenger creates an overlay to let users access the received messages conveniently

Overlay



Youtube creates an overlay to play the video while letting users interact with other applications simultaneously

Overlay Attack

- Overlays are widely abused by malware to launch attacks

Overlay Attack

- Overlays are widely abused by malware to launch attacks
  - Steal private information by monitoring user input

Overlay Attack

- Overlays are widely abused by malware to launch attacks
  - Steal private information by monitoring user input
  - Lure users to grant consent for sensitive operations

Overlay Attack

- Overlays are widely abused by malware to launch attacks
  - Steal private information by monitoring user input
  - Lure users to grant consent for sensitive operations

Overlay

- Case 1

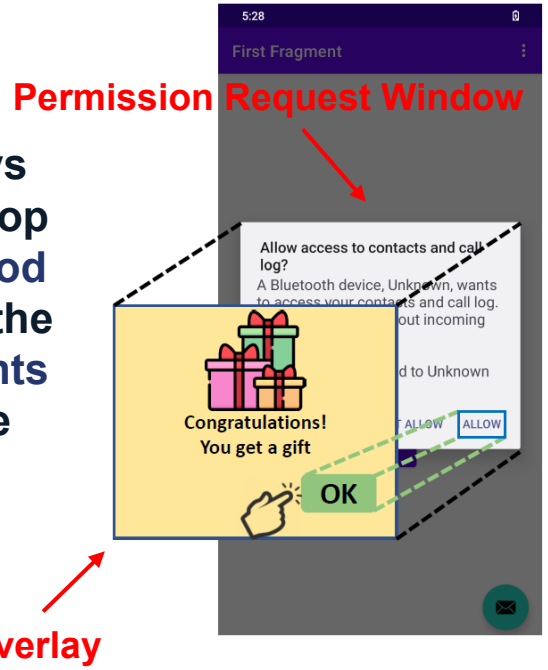
Malicious overlays are rendered on top of the input method to eavesdrop on the user's touch events to steal username and password

Overlay Attack

- Overlays are widely abused by malware to launch attacks
  - Steal private information by monitoring user input
  - Lure users to grant consent for sensitive operations

- **Case 1**

Malicious overlays are rendered on top of the input method to eavesdrop on the user's touch events to steal username and password



- **Case 2**

A malicious overlay is rendered on top of the system's settings app's permission request window, deceiving users into clicking on "Allow" button to grant the sensitive permission

System Apps

- System apps implement security-sensitive functionalities
  - Ask for consent before conducting sensitive operations

System Apps

- System apps implement security-sensitive functionalities
  - Ask for consent before conducting sensitive operations
  - Need to be protected against overlay attack

System Apps

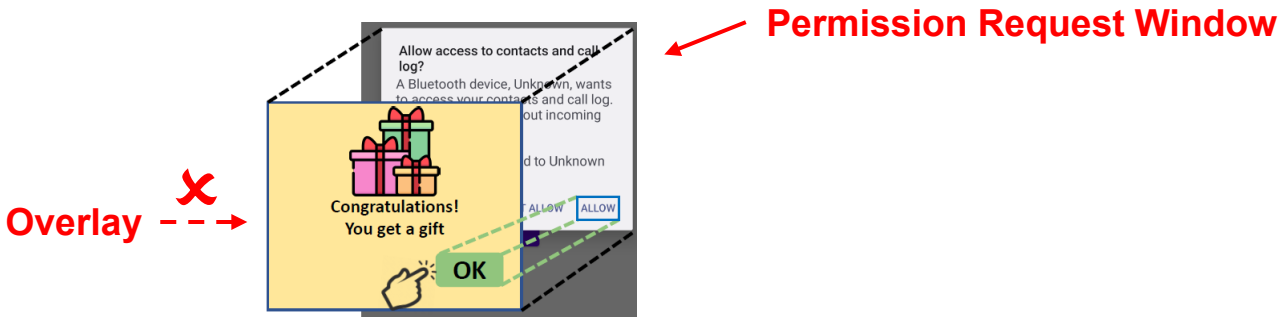
- System apps implement security-sensitive functionalities
  - Ask for consent before conducting sensitive operations
  - Need to be protected against overlay attack
- System provides a dedicated window flag namely SYSTEM _FLAG_ HIDE_NON_SYSTEM_OVERLAY_WINDOWS
  - Introduced in recently released systems Android 10.0~13.0

System Apps

- System apps implement security-sensitive functionalities
  - Ask for consent before conducting sensitive operations
  - Need to be protected against overlay attack
- System provides a dedicated window flag namely SYSTEM _FLAG_ HIDE_NON_SYSTEM_OVERLAY_WINDOWS
  - Introduced in recently released systems Android 10.0~13.0
  - System apps' windows can enable HNSOW to prevent overlays created by normal apps from drawing over them

System Apps

- System apps implement security-sensitive functionalities
  - Ask for consent before conducting sensitive operations
  - Need to be protected against overlay attack
- System provides a dedicated window flag namely SYSTEM _FLAG_ HIDE_NON_SYSTEM_OVERLAY_WINDOWS
  - Introduced in recently released systems Android 10.0~13.0
  - System apps' windows can enable HNSOW to prevent overlays created by normal apps from drawing over them



Unprotected System Apps

- **Unprotected system apps are prevalent**

Unprotected System Apps

- Unprotected system apps are prevalent
  - Google is constantly applying patches to enable HNSOW

Unprotected System Apps

- **Unprotected system apps are prevalent**
  - **Google is constantly applying patches to enable HNSOW**
  - **A series of vulnerabilities of missing protection against overlay attack in Android system apps have been exposed**

CVE Number	System App (Package Name)	Activity
CVE-2022-20212	com.android.settings	RequestToggleWifiActivity
CVE-2021-1016	com.android.systemui	UsbPermissionActivity
CVE-2021-0992	com.android.settings	PaymentDefaultDialog
CVE-2021-0538	com.android.phone	EmergencyCallbackModeExitDialog
CVE-2021-0523	com.android.settings	WifiScanModeActivity
CVE-2021-0391	android	ChooseTypeAndAccountActivity
CVE-2021-0333	com.android.settings	BluetoothPermissionActivity
CVE-2021-0314	com.android.packageinstaller	UninstallerActivity
CVE-2020-0394	com.android.settings	BluetoothPairingDialog
CVE-2020-0015	com.android.certinstaller	CertInstaller

Unprotected System Apps

- **Unprotected system apps are prevalent**
  - Google is **constantly** applying patches to enable HNSOW
  - A series of vulnerabilities of missing protection against overlay attack in Android system apps have been **exposed**

CVE Number	System App (Package Name)	Activity
CVE-2022-20212	com.android.settings	RequestToggleWifiActivity
CVE-2021-1016	com.android.systemui	UsbPermissionActivity
CVE-2021-0992	com.android.settings	PaymentDefaultDialog
CVE-2021-0538	com.android.phone	EmergencyCallbackModeExitDialog
CVE-2021-0523	com.android.settings	WifiScanModeActivity
CVE-2021-0391	android	ChooseTypeAndAccountActivity
CVE-2021-0333	com.android.settings	BluetoothPermissionActivity
CVE-2021-0314	com.android.packageinstaller	UninstallerActivity
CVE-2020-0394	com.android.settings	BluetoothPairingDialog
CVE-2020-0015	com.android.certinstaller	CertInstaller

- **CVE-2021-0333 BluetoothPermissionActivity in the system settings app was left unprotected**

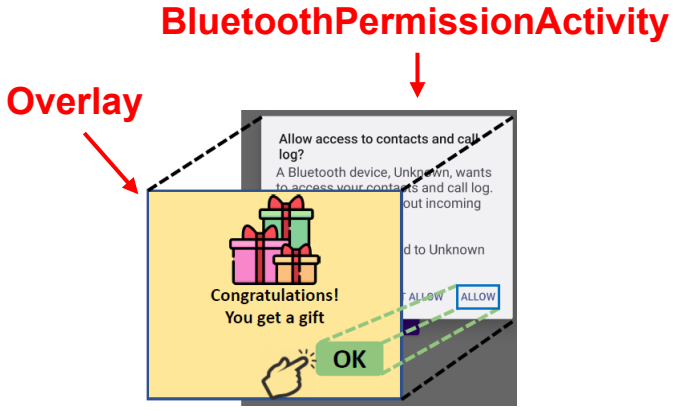
Unprotected System Apps

- **Unprotected system apps are prevalent**
  - Google is **constantly** applying patches to enable HNSOW
  - A series of vulnerabilities of missing protection against overlay attack in Android system apps have been **exposed**

CVE Number	System App (Package Name)	Activity
CVE-2022-20212	com.android.settings	RequestToggleWifiActivity
CVE-2021-1016	com.android.systemui	UsbPermissionActivity
CVE-2021-0992	com.android.settings	PaymentDefaultDialog
CVE-2021-0538	com.android.phone	EmergencyCallbackModeExitDialog
CVE-2021-0523	com.android.settings	WifiScanModeActivity
CVE-2021-0391	android	ChooseTypeAndAccountActivity
CVE-2021-0333	com.android.settings	BluetoothPermissionActivity
CVE-2021-0314	com.android.packageinstaller	UninstallerActivity
CVE-2020-0394	com.android.settings	BluetoothPairingDialog
CVE-2020-0015	com.android.certinstaller	CertInstaller

- **CVE-2021-0333**

BluetoothPermissionActivity in the system settings app was left unprotected



Existing Vague Guideline

- Google's documentation[1] provides a vague guidance
  - The window for granting permission.
  - The window for approving app installation.
  - The window for showing a persistent sensor icon or equivalent privacy-sensitive notification.

[1] https://bughunters.google.com/learn/invalid-reports/android-platform/5148417640366080/bugs-with-negligible-security-impact#tapjacking-overlay-system_alert_window-vulnerability-on-a-non-security-critical-screen

Existing Vague Guideline

- Google's documentation provides a vague guidance
- The vague guidance misses a large portion of windows requiring protection against overlay attack

Existing Vague Guideline

- Google's documentation provides a vague guidance
- The vague guidance misses a large portion of windows requiring protection against overlay attack

CVE Number	System App (Package Name)	Activity
CVE-2022-20212	com.android.settings	RequestToggleWifiActivity
CVE-2021-1016	com.android.systemui	UsbPermissionActivity
CVE-2021-0992	com.android.settings	PaymentDefaultDialog
CVE-2021-0538	com.android.phone	EmergencyCallbackModeExitDialog
CVE-2021-0523	com.android.settings	WifiScanModeActivity
CVE-2021-0391	android	ChooseTypeAndAccountActivity
CVE-2021-0333	com.android.settings	BluetoothPermissionActivity
CVE-2021-0314	com.android.packageinstaller	UninstallerActivity
CVE-2020-0394	com.android.settings	BluetoothPairingDialog
CVE-2020-0015	com.android.certinstaller	CertInstaller

- **Only three vulnerabilities in the table are covered by the guideline**
  - **CVE-2021-1016**
  - **CVE-2021-0333**
  - **CVE-2021-0314**

Existing Vague Guideline

- Google's documentation provides a vague guidance
- The vague guidance misses a large portion of windows requiring protection against overlay attack

CVE Number	System App (Package Name)	Activity
CVE-2022-20212	com.android.settings	RequestToggleWifiActivity
CVE-2021-1016	com.android.systemui	UsbPermissionActivity
CVE-2021-0992	com.android.settings	PaymentDefaultDialog
CVE-2021-0538	com.android.phone	EmergencyCallbackModeExitDialog
CVE-2021-0523	com.android.settings	WifiScanModeActivity
CVE-2021-0391	android	ChooseTypeAndAccountActivity
CVE-2021-0333	com.android.settings	BluetoothPermissionActivity
CVE-2021-0314	com.android.packageinstaller	UninstallerActivity
CVE-2020-0394	com.android.settings	BluetoothPairingDialog
CVE-2020-0015	com.android.certinstaller	CertInstaller

- **Only three vulnerabilities in the table are covered by the guideline**
  - **CVE-2021-1016**
  - **CVE-2021-0333**
  - **CVE-2021-0314**
- **The remaining cases are non-compliant with the guidance**

Existing Vague Guideline

- Google's documentation provides a vague guidance
- The vague guidance misses a large portion of windows requiring protection against overlay attack

A proper guideline for determining which windows of system apps need protection against overlay attack is in urgent need

Existing Vague Guideline

- Google's documentation provides a vague guidance
- The vague guidance misses a large portion of windows requiring protection against overlay attack

A proper guideline for determining which windows of system apps need protection against overlay attack is in urgent need

⟹ A systematic approach to identifying unprotected windows is crucial for bolstering security measures against overlay attack

Proposed Proper Guideline

- Analyze the windows under protection to build guideline
  - Conducted on the official Android system AOSP

Proposed Proper Guideline

- Analyze the windows under protection to build guideline
  - Conducted on the official Android system AOSP
  - ① Find the protected system apps' windows

Proposed Proper Guideline

- Analyze the windows under protection to build guideline
  - Conducted on the official Android system AOSP
  - ① Find the protected system apps' windows
    - Analyze addSystemFlags, addPrivateFlags, or setPrivateFlags

```
1 public class GrantPermissionsActivity
2   protected void onCreate(Bundle b){
3     getWindow().addSystemFlags(
4       SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS);}
```

Proposed Proper Guideline

- Analyze the windows under protection to build guideline
  - Conducted on the official Android system AOSP
  - ① Find the protected system apps' windows
    - Analyze addSystemFlags, addPrivateFlags, or setPrivateFlags

```
1 public class GrantPermissionsActivity
2   protected void onCreate(Bundle b){
3     getWindow().addSystemFlags(
4       SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS);}
```

  - ② Study the features of these protected windows

Proposed Proper Guideline

- Analyze the windows under protection to build guideline
  - Conducted on the official Android system AOSP
  - ① Find the protected system apps' windows
    - Analyze addSystemFlags, addPrivateFlags, or setPrivateFlags

```
1 public class GrantPermissionsActivity
2   protected void onCreate(Bundle b){
3     getWindow().addSystemFlags(
4       SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS);}
```

  - ② Study the features of these protected windows
    - Understand windows' functionalities

Proposed Proper Guideline

- Analyze the windows under protection to build guideline
  - Conducted on the official Android system AOSP
  - ① Find the protected system apps' windows
    - Analyze addSystemFlags, addPrivateFlags, or setPrivateFlags

```
1 public class GrantPermissionsActivity
2   protected void onCreate(Bundle b){
3     getWindow().addSystemFlags(
4       SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS);}
```

  - ② Study the features of these protected windows
    - Understand windows' functionalities
    - Find reasons for developers to enable HNSOW

Proposed Proper Guideline

- Analyze the windows under protection to build guideline
  - Conducted on the official Android system AOSP
  - ① Find the protected system apps' windows
    - Analyze addSystemFlags, addPrivateFlags, or setPrivateFlags

```
1 public class GrantPermissionsActivity
2   protected void onCreate(Bundle b){
3     getWindow().addSystemFlags(
4       SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS);}
```

  - ② Study the features of these protected windows
    - Understand windows' functionalities
    - Find reasons for developers to enable HNSOW
  - ③ Summarize the common features of protected windows
    - From aspects of startability, functionality, and interactivity

Proposed Proper Guideline

- **Startability**
  - **Windows under protection can be launched in one step**

Proposed Proper Guideline

- **Startability**
  - Windows under protection can be launched in one step
  - Windows that can be directly launched (i.e., launched in one step) are more vulnerable to overlay attack

Proposed Proper Guideline

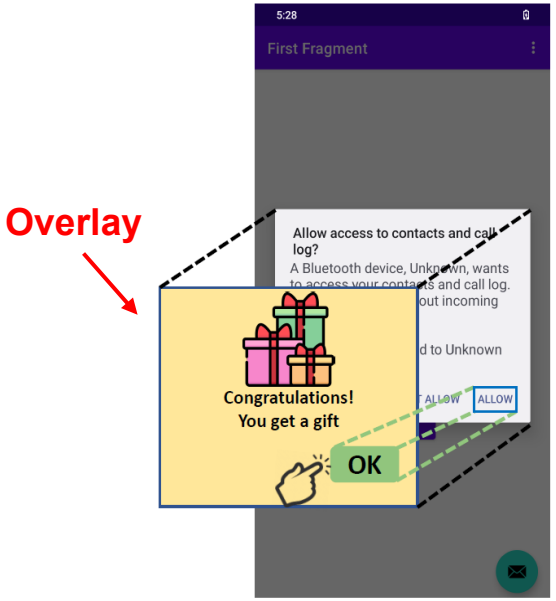
- **Startability**
  - Windows under protection can be launched in one step
  - Windows that can be directly launched (i.e., launched in one step) are more vulnerable to overlay attack

- **Example**

The activity BluetoothPermissionActivity of the system settings app, which asks users to grant permissions to the Bluetooth devices can be directly launched by malware

Proposed Proper Guideline

- **Startability**
  - Windows under protection can be launched in one step
  - Windows that can be directly launched (i.e., launched in one step) are more vulnerable to overlay attack



- **Example**

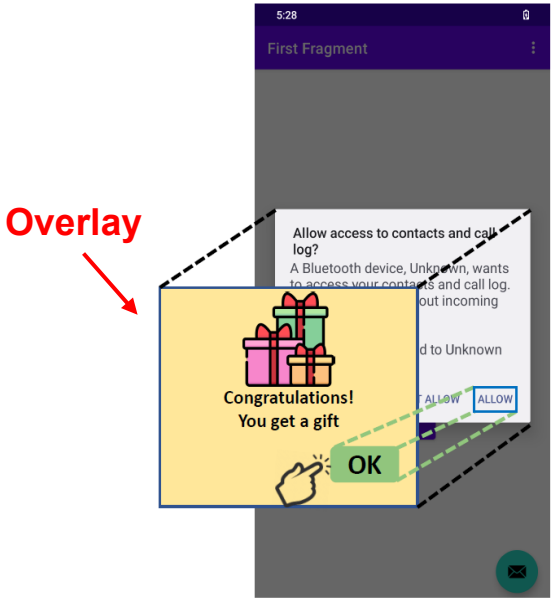
The activity BluetoothPermissionActivity of the system settings app, which asks users to grant permissions to the Bluetooth devices can be directly launched by malware

- **Attack**

Malware can draw an overlay on top of the activity to deceive users into clicking the "Allow" button to grant permission

Proposed Proper Guideline

- **Startability**
  - Windows under protection can be launched in one step
  - Windows that can be directly launched (i.e., launched in one step) are more vulnerable to overlay attack



- **Example**

The activity BluetoothPermissionActivity of the system settings app, which asks users to grant permissions to the Bluetooth devices can be directly launched by malware

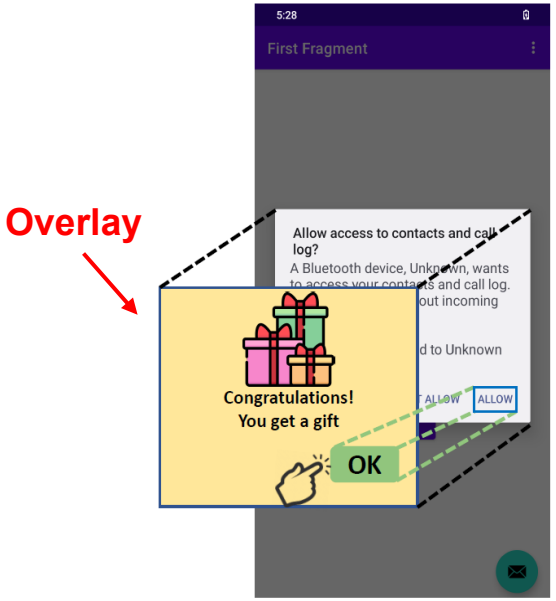
- **Attack**

Malware can draw an overlay on top of the activity to deceive users into clicking the "Allow" button to grant permission

⟹ Due to a lack of context information, users are unaware that they are interacting with BluetoothPermissionActivity

Proposed Proper Guideline

- **Startability**
  - Windows under protection can be launched in one step
  - Windows that can be directly launched (i.e., launched in one step) are more vulnerable to overlay attack



- **Example**

The activity BluetoothPermissionActivity of the system settings app, which asks users to grant permissions to the Bluetooth devices can be directly launched by malware

- **Attack**

Malware can draw an overlay on top of the activity to deceive users into clicking the "Allow" button to grant permission

⟹ Due to a lack of context information, users are unaware that they are interacting with BluetoothPermissionActivity

⟹ Android enables HNSOW in BluetoothPermissionActivity

Proposed Proper Guideline

- **Functionality**
  - **Windows under protection will perform sensitive operations**

Proposed Proper Guideline

- **Functionality**
  - Windows under protection will perform sensitive operations
  - Windows implementing sensitive functionalities typically require user consent to perform such operations

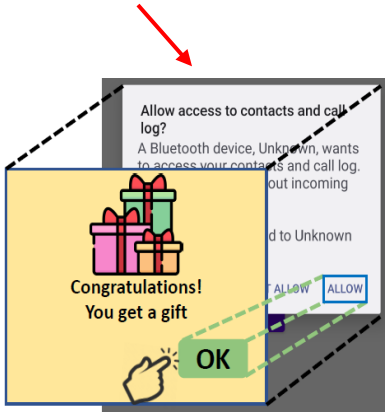
Proposed Proper Guideline

- **Functionality**
  - Windows under protection will perform sensitive operations
  - Windows implementing sensitive functionalities typically require user consent to perform such operations
  - Malicious overlay can deceive users into granting consent

Proposed Proper Guideline

- **Functionality**
  - Windows under protection will perform sensitive operations
  - Windows implementing sensitive functionalities typically require user consent to perform such operations
  - Malicious overlay can deceive users into granting consent

BluetoothPermissionActivity

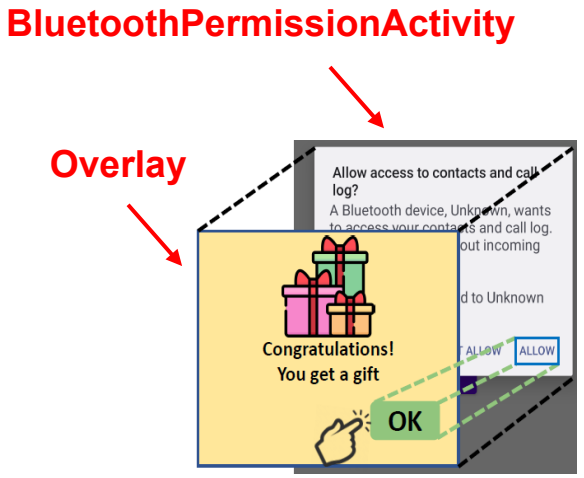


- **Example**

The activity BluetoothPermissionActivity implements sensitive function of granting permissions to Bluetooth devices

Proposed Proper Guideline

- **Functionality**
  - Windows under protection will perform sensitive operations
  - Windows implementing sensitive functionalities typically require user consent to perform such operations
  - Malicious overlay can deceive users into granting consent



- **Example**

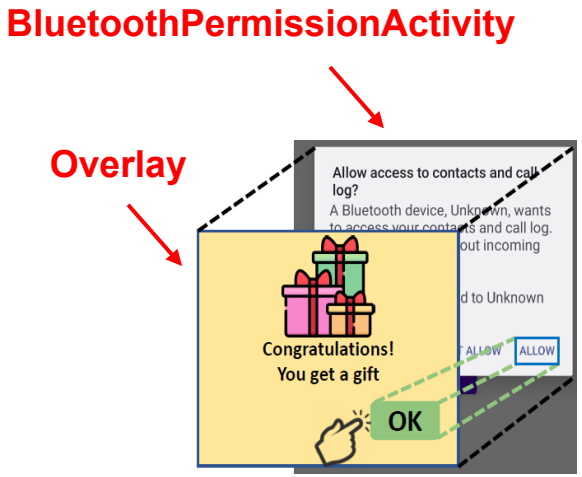
The activity BluetoothPermissionActivity implements sensitive function of granting permissions to Bluetooth devices

- **Attack**

Malware can draw an overlay on top of the activity to deceive users into clicking the "Allow" button to grant permissions

Proposed Proper Guideline

- **Functionality**
  - Windows under protection will perform sensitive operations
  - Windows implementing sensitive functionalities typically require user consent to perform such operations
  - Malicious overlay can deceive users into granting consent



- **Example**

The activity BluetoothPermissionActivity implements sensitive function of granting permissions to Bluetooth devices

- **Attack**

Malware can draw an overlay on top of the activity to deceive users into clicking the "Allow" button to grant permissions

→ Android enables HNSOW in BluetoothPermissionActivity

Proposed Proper Guideline

- **Interactivity**
  - **Sensitive functionalities of protected windows can normally be executed with no more than two user interactions**

Proposed Proper Guideline

- **Interactivity**
  - **Sensitive functionalities of protected windows can normally be executed with no more than two user interactions**
    - **Launch the window and trigger the sensitive functionality**

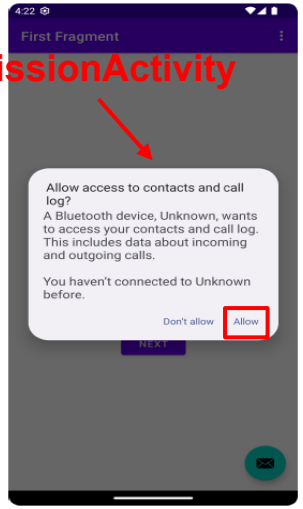
Proposed Proper Guideline

- Interactivity
  - Sensitive functionalities of protected windows can normally be executed with no more than two user interactions
    - Launch the window and trigger the sensitive functionality
  - Windows requiring fewer user interactions to execute sensitive functionalities are more vulnerable

Proposed Proper Guideline

- Interactivity
  - Sensitive functionalities of protected windows can normally be executed with no more than two user interactions
    - Launch the window and trigger the sensitive functionality
  - Windows requiring fewer user interactions to execute sensitive functionalities are more vulnerable

BluetoothPermissionActivity

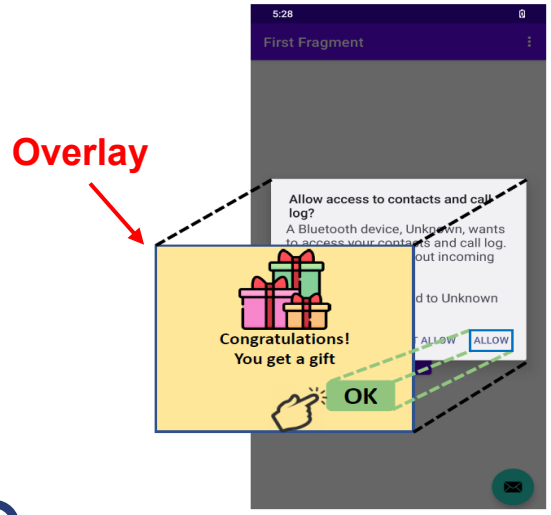


- Example

After launching BluetoothPermissionActivity, it only needs one click event to perform the sensitive permission granting operation

Proposed Proper Guideline

- **Interactivity**
  - Sensitive functionalities of protected windows can normally be executed with no more than two user interactions
    - Launch the window and trigger the sensitive functionality
  - Windows requiring fewer user interactions to execute sensitive functionalities are more vulnerable



- **Example**

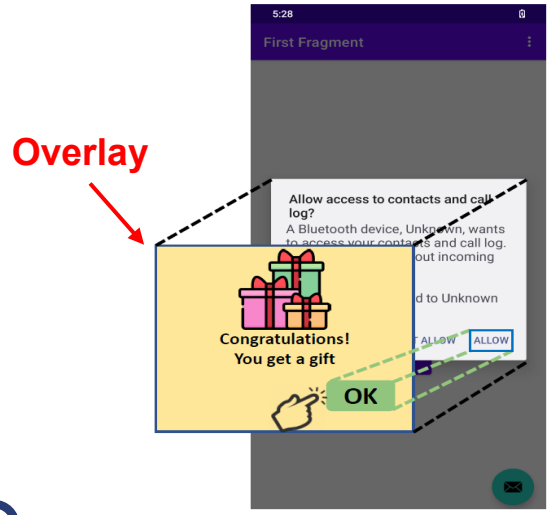
After launching BluetoothPermissionActivity, it only needs one click event to perform the sensitive permission granting operation

⇓

Malicious overlay can easily lure users into clicking the button

Proposed Proper Guideline

- **Interactivity**
  - Sensitive functionalities of protected windows can normally be executed with no more than two user interactions
    - Launch the window and trigger the sensitive functionality
  - Windows requiring fewer user interactions to execute sensitive functionalities are more vulnerable



- **Example**

After launching BluetoothPermissionActivity, it only needs one click event to perform the sensitive permission granting operation

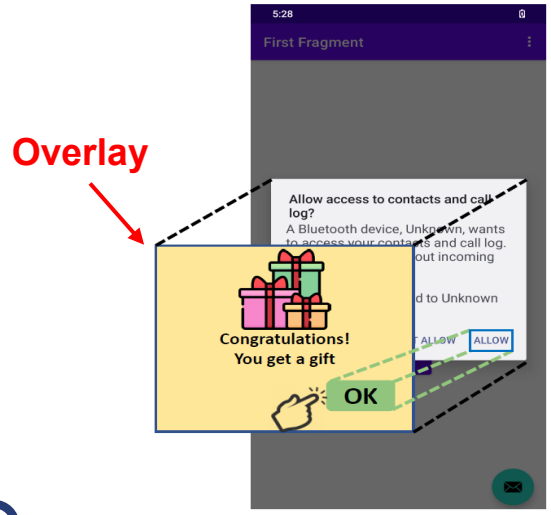
⇓

Malicious overlay can easily lure users into clicking the button

⟹ Users have limited context information and are challenging for them to know the consequences of such a simple click event

Proposed Proper Guideline

- Interactivity
  - Sensitive functionalities of protected windows can normally be executed with no more than two user interactions
    - Launch the window and trigger the sensitive functionality
  - Windows requiring fewer user interactions to execute sensitive functionalities are more vulnerable



- Example

After launching BluetoothPermissionActivity, it only needs one click event to perform the sensitive permission granting operation

⇓

Malicious overlay can easily lure users into clicking the button

⟹ Users have limited context information and are challenging for them to know the consequences of such a simple click event

⟹ Android enables HNSOW in BluetoothPermissionActivity

Proposed Proper Guideline

- Three criteria that serve as guidelines

Proposed Proper Guideline

- Three criteria that serve as guidelines
  - Criteria 1: One-Step Launch
    - The window can be directly launched

Proposed Proper Guideline

- Three criteria that serve as guidelines
  - Criteria 1: One-Step Launch
    - The window can be directly launched
  - Criteria 2: Sensitive Operation
    - The window implements security sensitive functionalities

Proposed Proper Guideline

- Three criteria that serve as guidelines
  - Criteria 1: One-Step Launch
    - The window can be directly launched
  - Criteria 2: Sensitive Operation
    - The window implements security sensitive functionalities
  - Criteria 3: Simplistic Interaction
    - Sensitive operations can be triggered by ≤ one interaction

Proposed Proper Guideline

- Three criteria that serve as guidelines
  - Criteria 1: One-Step Launch
    - The window can be directly launched
  - Criteria 2: Sensitive Operation
    - The window implements security sensitive functionalities
  - Criteria 3: Simplistic Interaction
    - Sensitive operations can be triggered by ≤ one interaction

A window satisfies all three criteria

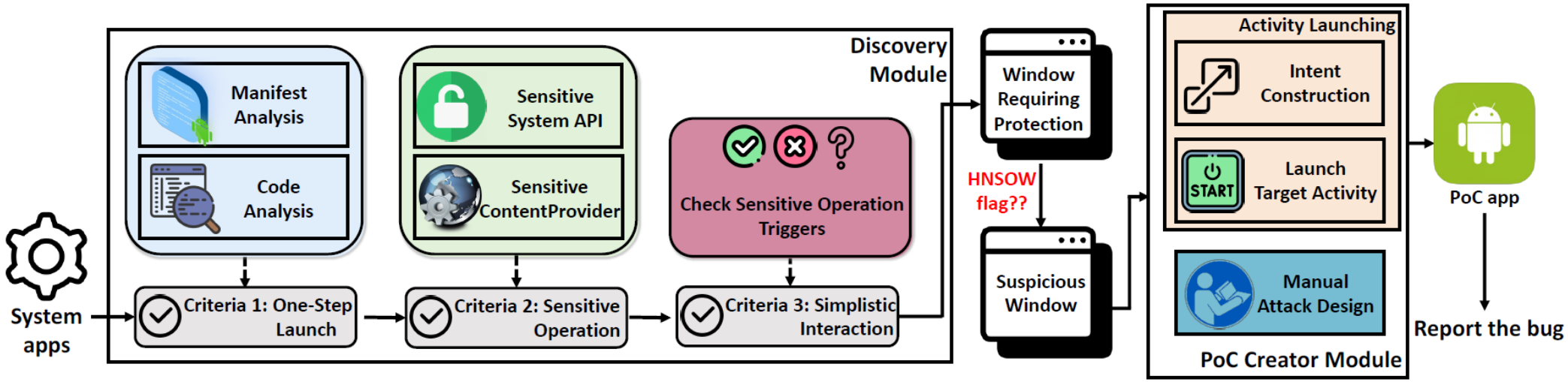
Need protection against overlay attack

OverlayChecker

- **Uncover the windows that miss protection**

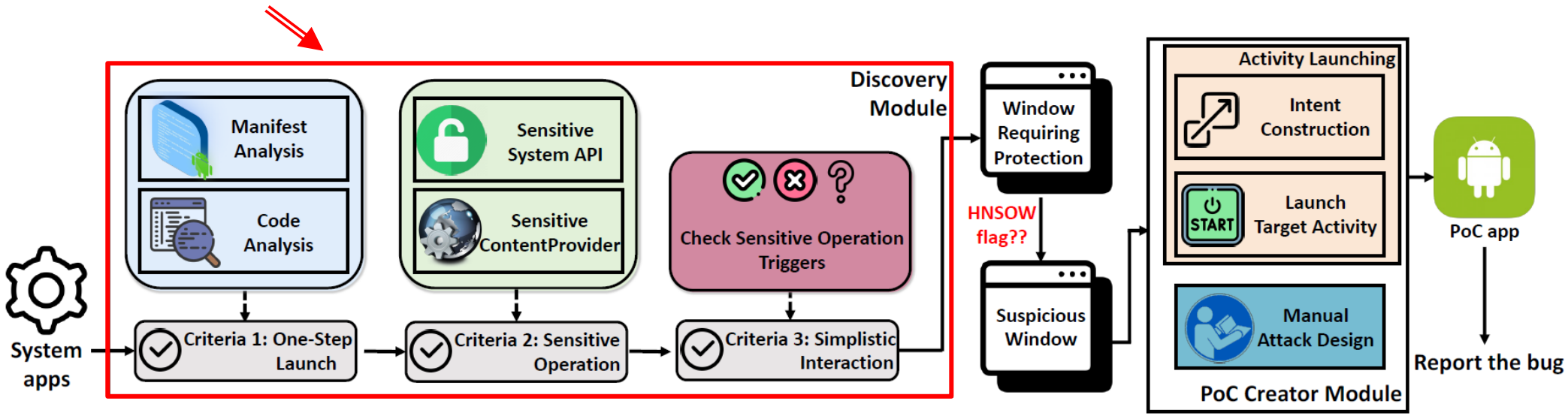
OverlayChecker

- Uncover the windows that miss protection
- It consists of discovery module and PoC creator module



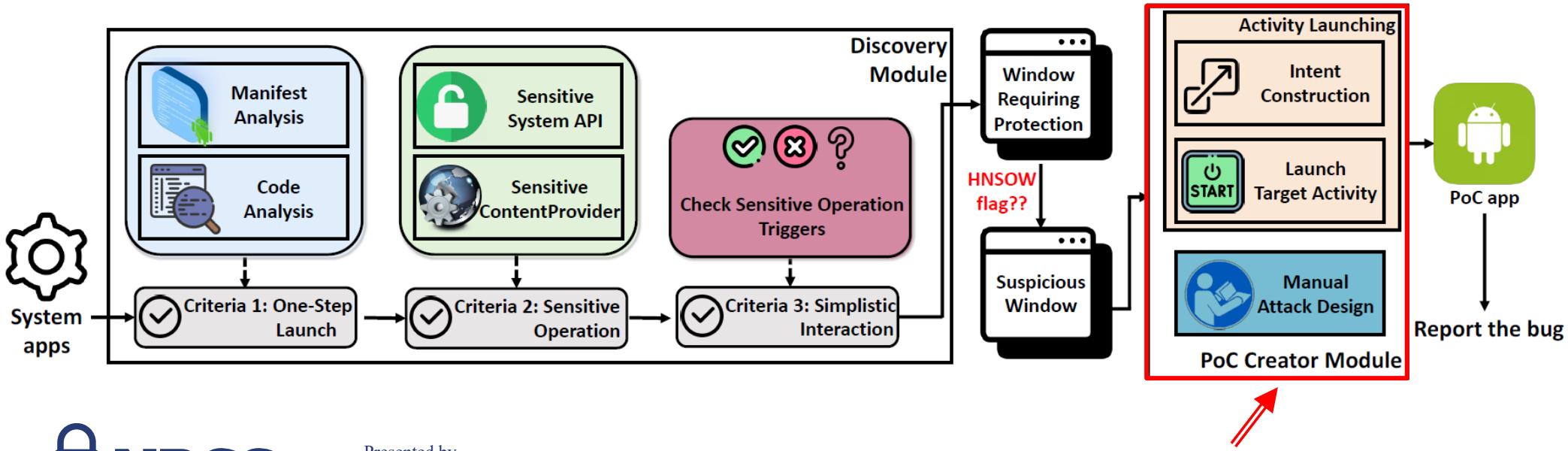
OverlayChecker

- Uncover the windows that miss protection
- It consists of discovery module and PoC creator module
  - Identify windows requiring protection against overlay attack

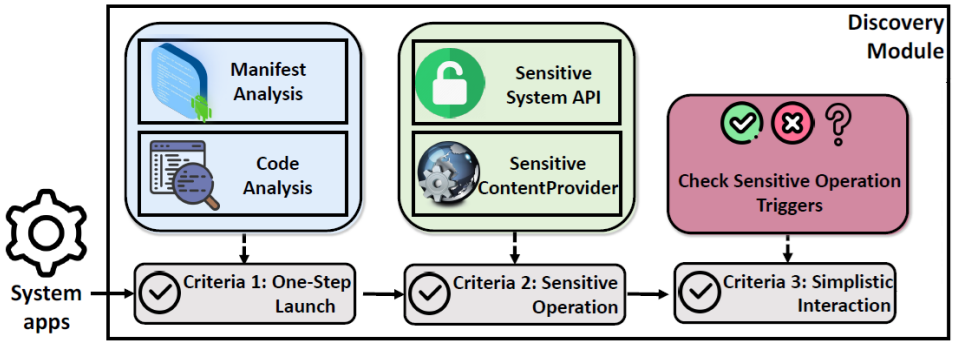


OverlayChecker

- Uncover the windows that miss protection
- It consists of discovery module and PoC creator module
  - Identify windows requiring protection against overlay attack
  - Assist in constructing PoC to confirm vulnerable windows

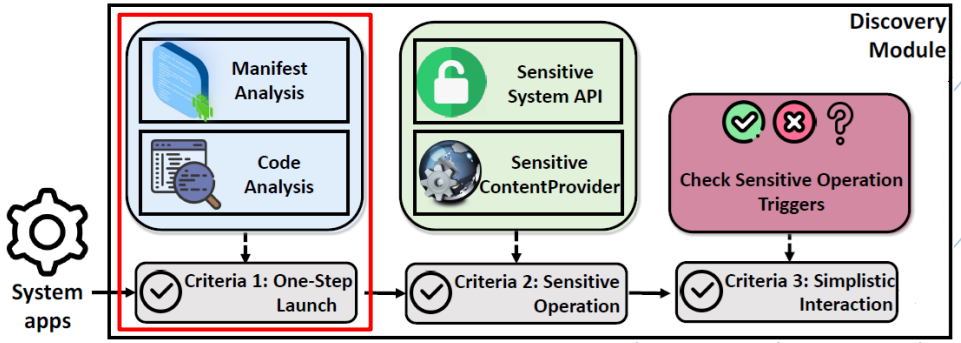


OverlayChecker

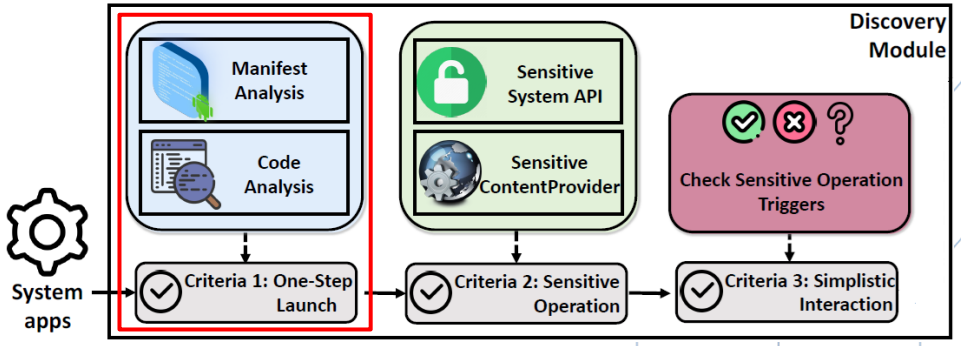


OverlayChecker

- **Criteria 1: One-Step Launch**
  - **Identify windows (activity, dialog) that can be directly launched**



OverlayChecker

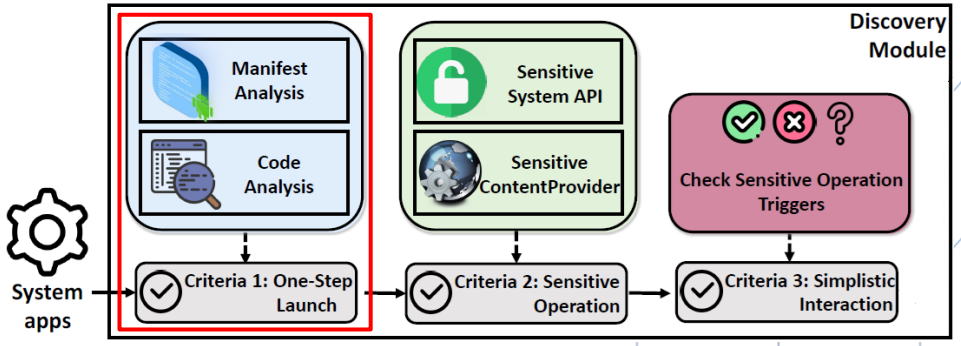


- **Criteria 1: One-Step Launch**
  - **Identify windows (activity, dialog) that can be directly launched**

① **Windows that can be directly launched through Intent objects**

⟹ **Analyze system apps' manifest files, especially "enable", "export" attributes**

OverlayChecker



- **Criteria 1: One-Step Launch**
  - **Identify windows (activity, dialog) that can be directly launched**

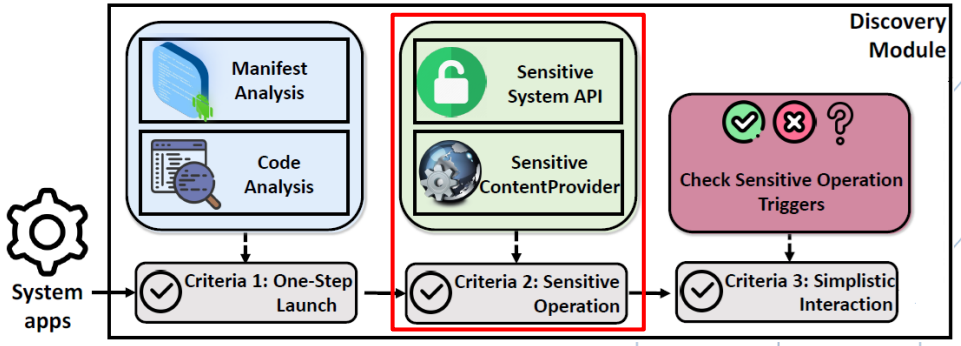
① Windows that can be directly launched through Intent objects

⟹ **Analyze system apps' manifest files, especially "enable", "export" attributes**

② Windows that can be directly launched by other components

⟹ **Analyze code of system apps and Android framework**

OverlayChecker



- **Criteria 1: One-Step Launch**
  - Identify windows (activity, dialog) that can be directly launched

① Windows that can be directly launched through Intent objects

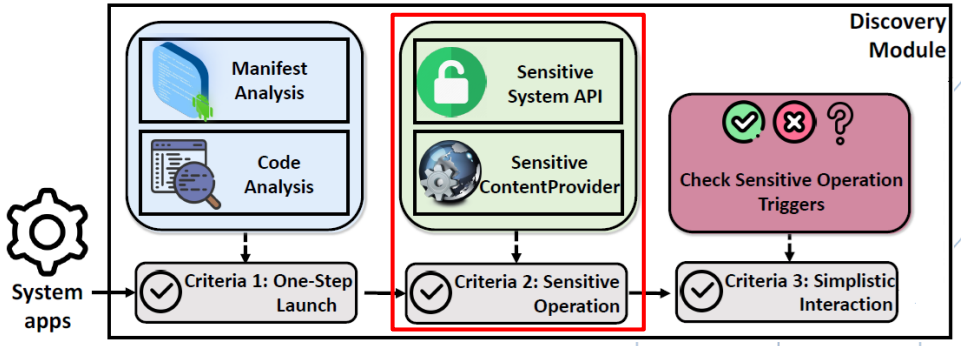
⟹ Analyze system apps' manifest files, especially "enable", "export" attributes

② Windows that can be directly launched by other components

⟹ Analyze code of system apps and Android framework

- **Criteria 2: Sensitive Operation**
  - Determine whether the windows call sensitive APIs or access sensitive content providers to conduct sensitive operations

OverlayChecker



- **Criteria 1: One-Step Launch**
  - Identify windows (activity, dialog) that can be directly launched

① Windows that can be directly launched through Intent objects

⟹ Analyze system apps' manifest files, especially "enable", "export" attributes

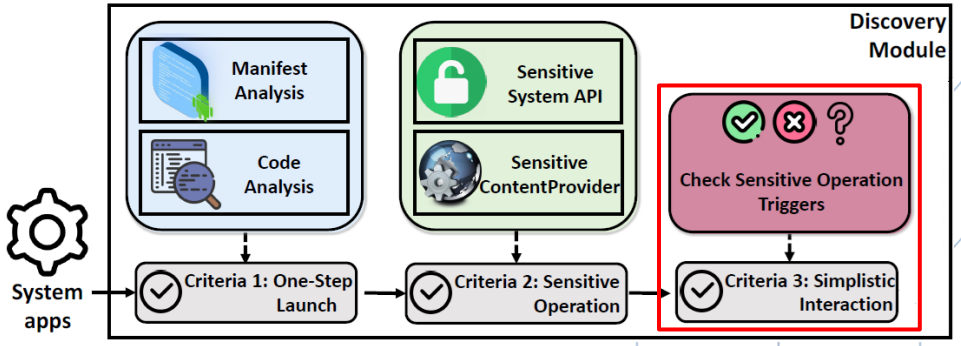
② Windows that can be directly launched by other components

⟹ Analyze code of system apps and Android framework

- **Criteria 2: Sensitive Operation**
  - Determine whether the windows call sensitive APIs or access sensitive content providers to conduct sensitive operations

⟹ Analyze event handlers since sensitive operations require user consent

OverlayChecker



- **Criteria 1: One-Step Launch**
  - Identify windows (activity, dialog) that can be directly launched

① Windows that can be directly launched through Intent objects

⟹ Analyze system apps' manifest files, especially "enable", "export" attributes

② Windows that can be directly launched by other components

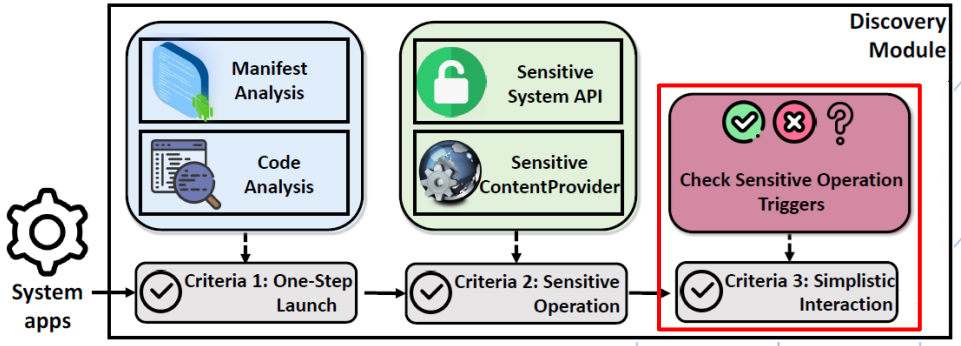
⟹ Analyze code of system apps and Android framework

- **Criteria 2: Sensitive Operation**
  - Determine whether the windows call sensitive APIs or access sensitive content providers to conduct sensitive operations

⟹ Analyze event handlers since sensitive operations require user consent

- **Criteria 3: Simplistic Interaction**
  - Determine whether sensitive operations require other user events

OverlayChecker



- **Criteria 1: One-Step Launch**
  - Identify windows (activity, dialog) that can be directly launched

① Windows that can be directly launched through Intent objects

⟹ Analyze system apps' manifest files, especially "enable", "export" attributes

② Windows that can be directly launched by other components

⟹ Analyze code of system apps and Android framework

- **Criteria 2: Sensitive Operation**
  - Determine whether the windows call sensitive APIs or access sensitive content providers to conduct sensitive operations

⟹ Analyze event handlers since sensitive operations require user consent

- **Criteria 3: Simplistic Interaction**
  - Determine whether sensitive operations require other user events

⟹ Analyze control/data dependency of sensitive operations

Is OverlayChecker reliable for identifying the windows of system apps require protection?

AOSP Version	# Protected	# Identified	# Missed
Android 10	27	27 (100.0%)	0 (0.0%)
Android 11	44	42 (95.5%)	2 (4.5%)
Android 12	60	56 (93.3%)	4 (6.7%)
Android 13	66	60 (90.9%)	6 (9.1%)

Is OverlayChecker reliable for identifying the windows of system apps require protection?

AOSP Version	# Protected	# Identified	# Missed
Android 10	27	27 (100.0%)	0 (0.0%)
Android 11	44	42 (95.5%)	2 (4.5%)
Android 12	60	56 (93.3%)	4 (6.7%)
Android 13	66	60 (90.9%)	6 (9.1%)

- **Most (more than 90%) of protected windows in AOSP Android 10~13 can be identified by OverlayChecker**

Is OverlayChecker reliable for identifying the windows of system apps require protection?

AOSP Version	# Protected	# Identified	# Missed
Android 10	27	27 (100.0%)	0 (0.0%)
Android 11	44	42 (95.5%)	2 (4.5%)
Android 12	60	56 (93.3%)	4 (6.7%)
Android 13	66	60 (90.9%)	6 (9.1%)

- **Most (more than 90%) of protected windows in AOSP Android 10~13 can be identified by OverlayChecker**

⟹ **OverlayChecker is reasonably reliable for identifying windows of system apps require protection**

Is OverlayChecker reliable for identifying the windows of system apps require protection?

AOSP Version	# Protected	# Identified	# Missed
Android 10	27	27 (100.0%)	0 (0.0%)
Android 11	44	42 (95.5%)	2 (4.5%)
Android 12	60	56 (93.3%)	4 (6.7%)
Android 13	66	60 (90.9%)	6 (9.1%)

- Most (more than 90%) of protected windows in AOSP Android 10~13 can be identified by OverlayChecker

⟹ OverlayChecker is reasonably reliable for identifying windows of system apps require protection

- 6 protected windows are missed by OverlayChecker

Is OverlayChecker reliable for identifying the windows of system apps require protection?

AOSP Version	# Protected	# Identified	# Missed
Android 10	27	27 (100.0%)	0 (0.0%)
Android 11	44	42 (95.5%)	2 (4.5%)
Android 12	60	56 (93.3%)	4 (6.7%)
Android 13	66	60 (90.9%)	6 (9.1%)

- Most (more than 90%) of protected windows in AOSP Android 10~13 can be identified by OverlayChecker

⟹ OverlayChecker is reasonably reliable for identifying windows of system apps require protection

- 6 protected windows are missed by OverlayChecker

⟹ They operate on file storing sensitive content instead of calling sensitive APIs or accessing sensitive content providers

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 12	Google	10
	Android 13		7

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 12	Google	10
	Android 13		7

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 10	Google	27
	Android 11		16
	Android 12		10
	Android 13		7

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 10	Google	27
	Android 11		16
	Android 12		10
	Android 13		7

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

- **Findings**

(1) Unprotected windows are gradually patched

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 10	Google	27
	Android 11		16
	Android 12		10
	Android 13		7

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

- **Findings**

(1) Unprotected windows are gradually patched
(2) Extra unprotected windows have introduced

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 10	Google	27
	Android 11		16
	Android 12		10
	Android 13		7

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

- **Findings**

(1) Unprotected windows are gradually patched
(2) Extra unprotected windows have introduced
(3) Unprotected windows still exist in latest Android

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 10	Google	27
	Android 11		16
	Android 12		10
	Android 13		7

Third-party Android Systems

System	Version	Vendor	# Unprotected
OneUI	Android 12	Samsung	26
OriginOS	Android 12	Vivo	22
MIUI	Android 12	Xiaomi	22
MagicUI	Android 12	Honor	14

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

- **Findings**

(1) Unprotected windows are gradually patched
(2) Extra unprotected windows have introduced
(3) Unprotected windows still exist in latest Android

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 10	Google	27
	Android 11		16
	Android 12		10
	Android 13		7

Third-party Android Systems

System	Version	Vendor	# Unprotected
OneUI	Android 12	Samsung	26
OriginOS	Android 12	Vivo	22
MIUI	Android 12	Xiaomi	22
MagicUI	Android 12	Honor	14

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

- **Findings**

(1) Unprotected windows are gradually patched
(2) Extra unprotected windows have introduced
(3) Unprotected windows still exist in latest Android

- **Identify 39 unprotected windows**

(1) All unprotected windows in AOSP Android 12 remain unprotected in third-party Android systems
(2) 2 CVEs are assigned with moderate severity
(3) Vivo rated the reported cases as high severity

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 10	Google	27
	Android 11		16
	Android 12		10
	Android 13		7

Third-party Android Systems

System	Version	Vendor	# Unprotected
OneUI	Android 12	Samsung	26
OriginOS	Android 12	Vivo	22
MIUI	Android 12	Xiaomi	22
MagicUI	Android 12	Honor	14

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

- **Findings**

(1) Unprotected windows are gradually patched
(2) Extra unprotected windows have introduced
(3) Unprotected windows still exist in latest Android

- **Identify 39 unprotected windows**

(1) All unprotected windows in AOSP Android 12 remain unprotected in third-party Android systems
(2) 2 CVEs are assigned with moderate severity
(3) Vivo rated the reported cases as high severity

- **Findings**

(1) More unprotected windows are found in third-party commercial Android systems

Can OverlayChecker uncover system apps' windows missing protection?

Official Android Systems

System	Version	Vendor	# Unprotected
AOSP	Android 10	Google	27
	Android 11		16
	Android 12		10
	Android 13		7

Third-party Android Systems

System	Version	Vendor	# Unprotected
OneUI	Android 12	Samsung	26
OriginOS	Android 12	Vivo	22
MIUI	Android 12	Xiaomi	22
MagicUI	Android 12	Honor	14

- **Identify 10 unprotected windows**

(1) All in Android 12, 7 in Android 13
(2) 3 CVEs are assigned with high severity

- **Findings**

(1) Unprotected windows are gradually patched
(2) Extra unprotected windows have introduced
(3) Unprotected windows still exist in latest Android

- **Identify 39 unprotected windows**

(1) All unprotected windows in AOSP Android 12 remain unprotected in third-party Android systems
(2) 2 CVEs are assigned with moderate severity
(3) Vivo rated the reported cases as high severity

- **Findings**

(1) More unprotected windows are found in third-party commercial Android systems
(2) Mobile vendors fail to promptly apply Google's security patches to their customized systems

Summary and Future Work

Summary and Future Work

- ✓ **Systematically study the vulnerability of missing protection against overlay attack in windows of Android system apps**

Summary and Future Work

- ✓ Systematically study the vulnerability of missing protection against overlay attack in windows of Android system apps
- ✓ Summarize criteria for determining whether a system app's window needs protection against overlay attack

Summary and Future Work

- ✓ Systematically study the vulnerability of missing protection against overlay attack in windows of Android system apps
- ✓ Summarize criteria for determining whether a system app's window needs protection against overlay attack
- ✓ Design OverlayChecker to uncover unprotected windows

Summary and Future Work

- ✓ Systematically study the vulnerability of missing protection against overlay attack in windows of Android system apps
- ✓ Summarize criteria for determining whether a system app's window needs protection against overlay attack
- ✓ Design OverlayChecker to uncover unprotected windows
- ✓ OverlayChecker finds 49 unprotected windows, leading to 5 CVEs and three of them are rated as High severity

Summary and Future Work

- ✓ Systematically study the vulnerability of missing protection against overlay attack in windows of Android system apps
- ✓ Summarize criteria for determining whether a system app's window needs protection against overlay attack
- ✓ Design OverlayChecker to uncover unprotected windows
- ✓ OverlayChecker finds 49 unprotected windows, leading to 5 CVEs and three of them are rated as High severity

- ➢ Investigate the vulnerability of missing protection against overlay attack in windows of third-party Android apps

Thanks for listening!