WindowGuard: Systematic Protection of GUI System in Android

Chuangang Ren, Peng Liu, Sencun Zhu

Department of Computer Science and Engineering
The Pennsylvania State University
Android Graphic User Interface

• Android GUI greatly promotes user experience
• One of the most sophisticated sub-systems in Android
Android GUI Security

• However, Android GUI system has been plagued by a variety of attacks that compromise the integrity and availability of Android GUI system.

• We call them **GUI attacks**
GUI Integrity Breach

• Mobile phishing attack\textsuperscript{1,2}

\textsuperscript{1} Chen et al. USENIX’14
\textsuperscript{2} Android Trojan Svpeng
GUI Integrity Breach

- Mobile phishing attack (USENIX’14, Svpeng malware)
- Task hijacking attack (USENIX’15)
GUI Integrity Breach

- Mobile phishing attack (USENIX’14, Svpeng malware)
- Task hijacking attack (USENIX’15)
- Tapjacking attack tricks user perform undesirable actions \(^4\)

\(^4\)Blackhat 12
GUI Availability Breach

- Ransomware migrates to mobile environment\(^1\), infecting 900K user devices within 2 years
- Adware repeatedly presents unwanted (sometime “uncloseable”) ad windows\(^2\)

\(^1\)Ransomware Police Locker

\(^2\)Rastogi, NDSS’16
Serious Security Threats

Σεριούσ τηρεάτο
Existing Defense

• Google has taken steps to remedy the security problems in newer Android versions
  ▪ Add security attributes to GUI components, e.g. `setFilterTouchesWhenObscured`
  ▪ Require explicit user consent when using certain permissions

• Challenges: adoption of the security features takes time
  ▪ Compatibility issues for existing functionalities
  ▪ Older devices or apps are vulnerable
Existing Defense

• Bianchi et al. (Oakland’15) proposes a two layer defense
  ▪ An app vetting process based on static analysis
  ▪ On-device defense mechanism

Ransomware: FBILock-A

App Locker
Contributions

• We systematically scrutinize the security implication of Android GUI system and find the root cause of GUI attacks

• We propose a new UI integrity model for Android - Android Window Integrity (AWI)

• We create WindowGuard – an implementation of AWI that protects user devices from all known GUI attacks
Building Blocks of GUI System

- **Activity:**
  - An app component that provides GUI to the user

- **Window:**
  - Conceptually, a visual area on screen that shows the GUI
  - A container to hold all GUI components

- An activity must include a window
Window Management

Important Notions:
Window stack, Window Z-order, Window visibility, Window Token
Existing security mechanisms:

- App sandboxing, protected by Linux UID
- Window token
- Permission

**Security Risk:** an user session is beyond the scope of existing security mechanism protections
• An **user session or activity session** is a sequence of activities that user has interacted in a particular job
  
  • Activities in an user session may come from different apps
  
  • Great flexibility that allows apps to control activity and window behaviors

Two activity sessions:

• Launcher -> A1 -> A2
• Launcher -> B1 -> C2

No security guarantees
• **Key principle:** no app has permission to perform any operations that would adversely affect other app’s activity session

  • **Display owner:** display owner is the app of focused activity. Display owner “owns” the screen. Display owner and the focused user session is protected by AWI.
Android Window Integrity (AWI)

AWI is composed of three legitimacy:

- Legitimacy of activity session
- Legitimacy of future windows
- Legitimacy of existing windows
Legitimacy of Activity Session

Criteria: focused activity session should always be consistent with the back stacks in AMS

Formally:

$$\exists\{bs_1^*, bs_2^*, ..., bs_n^*\} \subseteq \beta : sf_{fg} = (bs_1^* || bs_2^* ||, ..., || bs_n^*)$$

- $bs_i^*$: a back stack (a sequence of activities)
- $\beta$: all back stacks in the system
- $sf_{fg}$: focused activity session (a sequence of activities)
Valid System State

A valid example:

L → B1 → B2

L → A1 → A2

Tasks:

Activity sequence:

Activity Sessions:
A task hijacking example:

Activity Stacks:

Activity Sequence:

Activity Sessions:

Focused Activity

Phishing Activity
WindowGuard

• We implement AWI as a Xposed module – *WindowGuard*, by hooking various framework components in Android GUI system

  • WindowGuard prompts the user for the final decision once a security violation occurs. This design meets the diverse needs of users and app developers in the Android ecosystem.

  • 5 security features, such as integrity of activity session, legitimacy of windows start/resume, etc.
Tapjacking Attack Example

An Android malware (BankRob) example:

Device Admin Request Confirmation Dialog (a)

Tapjacking Overlay Window (b)

WindowGuard Security Alert (c)
### Effectiveness

<table>
<thead>
<tr>
<th>Attack Vectors</th>
<th>Consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Stolen</td>
</tr>
<tr>
<td>UI Interception</td>
<td>![Skull]</td>
</tr>
<tr>
<td>Tap-jacking</td>
<td>![Skull]</td>
</tr>
<tr>
<td>Back-button Hijacking</td>
<td>![Skull]</td>
</tr>
<tr>
<td>Activity launch hijacking</td>
<td>![Skull]</td>
</tr>
<tr>
<td>User Monitoring Attack</td>
<td>![Skull]</td>
</tr>
<tr>
<td>Adware</td>
<td>![Skull]</td>
</tr>
</tbody>
</table>

*WindowGuard can defeat all known GUI attacks.*
Usability

- We evaluate the usability by automatically exercising each of 12,060 most popular Google Play apps for 5 minutes on devices with WindowGuard enabled.

<table>
<thead>
<tr>
<th>Security Feature</th>
<th>Alert Msg</th>
<th># of Apps</th>
<th>% of Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Session Legitimacy</td>
<td>T, N</td>
<td>12</td>
<td>0.10</td>
</tr>
<tr>
<td>New Window Access Control</td>
<td>D</td>
<td>39</td>
<td>0.32</td>
</tr>
<tr>
<td>Existing Window Legitimacy</td>
<td>T, N</td>
<td>14</td>
<td>0.12</td>
</tr>
<tr>
<td>New Activity Control</td>
<td>D</td>
<td>69</td>
<td>0.57</td>
</tr>
<tr>
<td>Activity Resume Legitimacy</td>
<td>D</td>
<td>11</td>
<td>0.09</td>
</tr>
<tr>
<td>Any Feature(s)</td>
<td></td>
<td>124</td>
<td>1.03</td>
</tr>
</tbody>
</table>

- Only 1% apps triggers security alert
- Among those apps that trigger security alert, 62.5% triggers security alert only once
Limitation

• WindowGuard introduces 1% of false positives

• The flexibility of letting user make the final security decision may introduce false negatives.

• The current implementation of WindowGuard is based on Xposed, which can only be used on rooted devices.
Conclusion

• We systematically scrutinize the security implication of Android GUI system

• We propose a new UI integrity model – Android Window Integrity model

• We implement WindowGuard, which is able to effectively defeat all known GUI attacks
Thank you!

(Contact: chuangang.ren@gmail.com)
Back-up Slides
Existing Defense

• Bianchi et al. (Oakland’15) proposes a two layer defense
  ▪ An app vetting process based on static analysis

Ransomware: FBIlock-A

App Locker
Challenges

• Challenges of existing on-device defense
  ▪ Negative impact on user experience
  ▪ Low detection accuracy (max. 76% in an user study)
  ▪ Only capable of defending against GUI confusion attack
Put Everything Together

משלה addView()}

 AMS

 Task A

 A2

 A1

 Task B

 B1

 Task L

 Launcher

 Apps

 System Services

 WMS

 Permission Check

 Navigation Bar Win
 Status Bar Win
 Keyguard Win
 Toast Win
 A2 Sub Win
 A2 Win
 A1 Win
 Input Win
 B1 Win
 Launcher Win
 Wallpaper Win

 Window stack

 Mobile Display

 This is a Toast message

 Progress Dialog

 Please wait...

 CUSTOM VIEW

 CUSTOM VIEW IN SCROLL #1

 CUSTOM VIEW IN SCROLL #2

 CUSTOM VIEW IN SCROLL #3
Existing Defense

- Bianchi et al. (Oakland’15) proposes a two layer defense
  - An app vetting process based on static analysis
  - On-device defense mechanism

Extended Validation green address bar in modern browsers

App identity indicator in Android
Legitimacy of Windows

Legitimacy of Future Windows

Criteria: the principal that launches (or resumes) a window must be either the display owner app or a white list of principals (e.g., system UI).

Legitimacy of Existing Windows

Criteria: no existing windows should be placed on top of the display owner’s window, unless it is from a white list of principals.
Performance

• We evaluate the performance of WindowGuard by a comparison study.

• We generate the same sequence of 5000 user events to 10 app w/ and w/o WindowGuard installed

• On average, Windowguard incurs 0.8% performance overhead.