Oral Exam: Apposcopy (Semantics-Based Malware Detection)

Group 12 Presentation Script

0:00–0:45 — The Problem

- **Context:** Android malware is rapidly evolving, often stealing private user data.
- **The Gap:** Traditional detection methods have significant flaws:
  - *Taint Analysis:* Cannot distinguish between malicious theft and legitimate functionality (e.g., an email app *must* send data to the internet).
  - *Signature-Based (Syntactic):* Relies on byte-level patterns. It is easily defeated by simple obfuscation techniques like renaming or code reordering.
- **Goal:** To detect malware based on **semantics** (behavior) rather than syntax, effectively identifying malware families even when obfuscated.

0:45–2:45 — The Solution: Apposcopy

The authors propose matching high-level signatures against static analysis results.

1. **Core Concept:** Malware is defined by *what it does.* The system analyzes:

   $$\text{Malware} = \text{Control Flow (ICC)} + \text{Data Flow (Taint)}$$

2. **Inter-Component Call Graph (ICCG):** They build a graph where nodes are components (Activities, Services, Receivers) and edges represent communication (Intents). This abstracts away local code changes.
3. **Signature Language (Datalog):** Malware families are defined using logical predicates:
   - `icc(p, q)`: Component P sends an Intent to Q.
   - `flow(s, source, s, sink)`: Sensitive data flows from a specific source to a sink.
4. **Example (GoldDream Family):** The signature looks for a specific behavioral pattern: A Receiver listening for SMS events → Starts a Service → That Service leaks DeviceID to the Internet.

2:45–4:00 — Evaluation

Tested on the **Android Malware Genome Project** and Google Play.

- **Accuracy:** Achieved 90% detection accuracy overall.
  - *Success:* Excellent detection of families like *DroidKungFu* and *GoldDream.*

  - *Failure:* Performed poorly on the *BaseBridge* family (38% accuracy) because it uses **dynamic code loading**, which static analysis cannot see.

- **Resilience:** When testing obfuscated malware (using ProGuard and encryption), Apposcopy maintained high detection rates while commercial AVs (like AVG and Symantec) failed significantly.

- **Comparison:** Outperformed **Kirin** (a permission-based tool), which had a 48% false negative rate compared to Apposcopy's 10% on the malware set.

4:00–5:00 — Critique & Conclusion

- **Strengths:**
  - **Semantic Detection:** Combining ICC and Taint analysis drastically lowers false positives compared to using either method alone.
  - **Obfuscation Resilience:** The high-level graph approach makes it robust against standard renaming attacks.

- **Weaknesses:**
  - **Dynamic Code Loading:** This remains the "Achilles Heel." Apposcopy cannot analyze code that is downloaded at runtime.
  - **Known Families Only:** It is not a zero-day detector. It requires a pre-written Datalog signature for a specific family.

- **Verdict:** Apposcopy is highly effective for vetting apps in a store pipeline against known threats, but must be paired with dynamic analysis to catch modern, dynamic-loading malware.