SECURITY READING GROUP

Schedule: talks.cs.umd.edu/lists/19

Today’s Talk: User Interaction and Permission Use on Android - Daniel Votipka

Upcoming Talks:
Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study - Kristopher Micinski
3450 A.V. Williams
22 Feb @ 1:00pm

Information Flow Security in Practical Systems - Limin Jia, CMU
3460 A.V. Williams
3 Mar @ 11:00am

Signup: http://ter.ps/d7k
User Interactions and Permission Use on Android

Daniel Votipka
University of Maryland
Android Permissions

• Permissions guard access to sensitive resources

• Per-app tokens that allow app to access data

• Authorized at install time

• Newer versions: on first use of resource
Android Permissions

• Permissions guard access to sensitive resources
• Per-app tokens that allow app to access data
• Authorized at install time
• Newer versions: on first use of resource
All permission systems make choices to balance invasiveness with transparency.
Integrate Permissions w/ UI

• UI deeply informs user’s mental model of behavior

• Hypothesis: Achieve better balance by integrating UI

• Complementary studies measure this
  
  • App study: determine when top apps access resources
  
  • User study: study how interaction affects expectation
App Study

• When do top apps access resources?
• Dynamic analysis / visualization
  • Show when access happens in context of app UI
• Assemble codebook to categorize patterns
• 150 top apps from Google Play
Dynamic analysis tool to classify resource use.

- Collects logs from instrumented app.
- Log method entries/exit via binary rewriting.
- Manually explore app to collect logs.
- Generates event graph from logs.

Diagram:
- Android App
  - Tester
  - Coder A
  - Binary Rewriting
  - Execution Log
  - Log Visualization
  - AppTracer
  - Resolve Differences
  - Resolved Uses
  - New Resource Uses
• Log UI-relevant and permissions-relevant calls
• Uses PScout, maps API calls to permissions
• Log method names and parameters / return values
• Bookkeeping info: threads, screenshots, etc…
• Extends Redexer, Android rewriter we developed

[Au et al. CCS-2012]
public class HomeActivity {
  public class DoTask {
    public void run() {
      Location l = LocManager.getLastLoc();
      // ...
      String n = TelManager.getLine1Number();
      // ...
    }
  }
}

public void onCreate(Bundle b) {
  // ...
  findCoffeeButton.setOnClickListener(
    new View.OnClickListener() {
      public void onClick(int id) {
        DoTaskThread t = new DoTaskThread();
        t.start();
      }
    });
}
public class HomeActivity {
    public class DoTask {
        public void run() {
            logEnt("run");
            logEnt("Loc...getLastLoc");
            Location l = LocManager.getLastLoc();
            logExit("Loc...getLastLoc", l);
            // ...
            logEnt("Tel...getLine1Number");
            String n = TelManager.getLine1Number();
            logExit("Tel...getLine1Number", n); }
        // ...
        logExit("run");
    }
    public void onCreate(Bundle b) {
        logEnt("HomeAc...onCreate",this,b);
        // ...
        findCoffeeButton.setOnClickListener(
            new View.OnClickListener() {
                public void onClick(int id) {
                    logEnt("onClick");
                    DoTaskThread t = new DoTaskThread();
                    t.start();
                }
            });
        // ...
        logExit("HomeActivity.onCreate");
    }
}
public class HomeActivity {
    public class DoTask {
        public void run() {
            logEnt("run");
            logEnt("Loc...getLastLoc");
            Location l = LocManager.getLastLoc();
            logExit("Loc...getLastLoc", l);
            // ...
            logEnt("Tel...getLine1Number");
            String n = TelManager.getLine1Number();
            logExit("Tel...getLine1Number", n); }
        // ...
        logExit("run");
    }
    public void onCreate(Bundle b) {
        logEnt("HomeAc...onCreate",this,b);
        // ...
        findCoffeeButton.setOnClickListener(
            new View.OnClickListener() {
                public void onClick(int id) {
                    logEnt("onClick");
                    DoTaskThread t = new DoTaskThread();
                    logThreadStart(t);
                    t.start();
                    logExit("onClick");
                }
            });
        logExit("HomeActivity.onCreate");
    }
}
public class HomeActivity {
    public class DoTask {
        public void run() {
            logEnt("run");
            logEnt("Loc...getLastLoc");
            Location l = LocManager.getLastLoc();
            logExit("Loc...getLastLoc", l);
            // ...
            logEnt("Tel...getLine1Number");
            String n = TelManager.getLine1Number();
            logExit("Tel...getLine1Number", n);  
        }
        // ...
        logExit("run");
    }
    public void onCreate(Bundle b) {
        logEnt("HomeAc...onCreate",this,b);
        // ...
        findCoffeeButton.setOnClickListener(
            new View.OnClickListener() {
                public void onClick(int id) {
                    logEnt("onClick");
                    DoTaskThread t = new DoTaskThread();
                    logThreadStart(t);
                    t.start();
                    logExit("onClick");
                }
            });
        logExit("run");
    }
    public void onStop() {
        logEnt("HomeAc...onStop");
        logExit("run");
    }
}
• Interprets log to assemble event graph of program

• Nodes are UI/permissions-relevant events
  • Identified by method names

• Edge between nodes when one happens before another
  • Also partial model of system (activities, services, etc.)

• Nodes are enclosed in boxes representing app activities
> HomeActivity.onCreate()
< HomeActivity.onCreate()
> onClick(id = "Find Coffee")
> Thread start(id = 323)
< onClick()
> (323) DoTask.run()
> (323) LocManager.getLastLoc()
< (323) LocManager.getLastLoc(Loc="...")
> (323) TelManager.getLine1Number()
< (323) LocManager.getLine1Number("...")
< (323) DoTask.run()
> HomeActivity.onStop()
< HomeActivity.onStop()
> HomeActivity.onCreate()
< HomeActivity.onCreate()
> onClick(id = "Find Coffee")
> Thread start(id = 323)
< onClick()
> (323) DoTask.run()
> (323) LocManager.getLastLoc()
< (323) LocManager.getLastLoc(Loc=<...>)
> (323) TelManager.getLine1Number()
< (323) LocManager.getLine1Number("...")
< (323) DoTask.run()
> HomeActivity.onStop()
< HomeActivity.onStop()
• Define classification system for resource uses

• Initial codebook based off knowledge of apps

• Refined codebook iteratively by coding sets of five apps (20 apps total)

• Six unique codes representing patterns in apps
Code Definitions

**Click**
- Directly after click on related UI element

**Page**
- Throughout the duration of a related activity
Code Definitions

**Startup**
- After startup but before first activity

**Bg-App**
- When app in foreground, but not directly because of related UI event

**Bg-Ext**
- In response to system event (app not necessarily on screen)
Code Definitions

**Uncertain**

- When AppTracer not precise enough for us to say
Study Mechanics

- Downloaded top 20 free-apps from 27 Google Play categories -> 503 unique apps

- Randomly selected 150 apps to test with the following exclusions:
  - Redexer failed to rewrite 48 apps
  - 23 apps would not run in modified form
  - 16 apps required accounts that could not be easily acquired (e.g. bank account)
Limitations

- Dynamic analysis does not cover all possible execution paths
- Redexer could miss pertinent method calls
- Imprecision introduced by AppTracer visualization
- We only test on popular apps
App Study Results
Sensitive resources mainly used interactively

Legitimate outliers (e.g., taking pic of intruder)
Mix interactive / non-interactive use

Frequently supported foreground use
Mostly non-interactive use

Devs believe less sensitive?

Hard to explain to users?
Mostly click-based interaction. Location used frequently for (e.g.,) maps.
User Expectation Survey

• Do user expectations align with patterns used?

• **H1.** Users are more likely to expect resource access with an interactive use pattern than without

• **H2.** The more apps use resources, the more likely users are to expect background uses

• **H3.** Users are more likely to expect resource accesses they have seen before
Blurb about what app does
User Action: Click, Notification, or Background (Clk,Bn,Bg)

Possibly show auth dialog (Could also occur at launch or never)

Note: this scenario about microphone
Likert scale questions
Asks about mic
Others for distraction
Implicitly measures Bg use for location
Ex: “which button would you press if you wanted to find a new coffee shop and add it to your favorites?”
Go back to home screen
Is Mic still expected? Measures Bg use after foreground use
# Conditions Studied

<table>
<thead>
<tr>
<th>App</th>
<th>Resource</th>
<th>Authorization</th>
<th>Int. Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Mic</td>
<td>First use</td>
<td>Clk-Clk</td>
</tr>
<tr>
<td>Fitness</td>
<td>Contacts</td>
<td>Launch</td>
<td>Clk-Bg</td>
</tr>
<tr>
<td>Location</td>
<td>Never</td>
<td>Bn-Bg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bg-Bg</td>
</tr>
</tbody>
</table>
## Conditions Studied

<table>
<thead>
<tr>
<th>App</th>
<th>Resource</th>
<th>Authorization</th>
<th>Int. Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Mic</td>
<td>First use</td>
<td>Clk-Clk</td>
</tr>
<tr>
<td>Fitness</td>
<td>Contacts</td>
<td>Launch</td>
<td>Clk-Bg</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>Never</td>
<td>Bn-Bg</td>
</tr>
</tbody>
</table>

Chose three resources we believe representative
### Conditions Studied

<table>
<thead>
<tr>
<th>App</th>
<th>Resource</th>
<th>Authorization</th>
<th>Int. Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Mic</td>
<td>First use</td>
<td>Clk-Clk</td>
</tr>
<tr>
<td>Fitness</td>
<td>Contacts</td>
<td>Launch</td>
<td>Clk-Bg</td>
</tr>
<tr>
<td>Location</td>
<td>Never</td>
<td>Bn-Bg</td>
<td>Bg-Bg</td>
</tr>
</tbody>
</table>

Only tested with Launch
## Conditions Studied

<table>
<thead>
<tr>
<th>App</th>
<th>Resource</th>
<th>Authorization</th>
<th>Int. Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Mic</td>
<td>First use</td>
<td>Clk-Clk</td>
</tr>
<tr>
<td><strong>Fitness</strong></td>
<td>Contacts</td>
<td>Launch</td>
<td>Clk-Bg</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>Never</td>
<td>Bn-Bg</td>
</tr>
</tbody>
</table>

Clk: Click, Bn: Button, Bg: Background
Study Mechanics

• Mechanical Turk survey of 961 users
• Each user saw one scenario
• Paid $1 for survey, median time of ~5 minutes
• Tested 42 different scenarios
  • Used round-robin assignment
Statistical Analysis

• Logistic regressions over Likert data

• Two regressions:
  
  • First access (H1: how does interaction affect expectation, H2: Does the number of apps using a resource affect expectation)
  
  • Second access (H3: how do prior accesses affect future expectation)

• Compared to baseline scenario, how much more likely are you to go up one ordinal point in scale?

  • Baseline scenario: Coffee, Mic, Never, Bg-Bg
Limitations

- We only use two mock apps
- Relatively short period between the first and second resource use
- Participants may not answer truthfully or may try to submit multiple responses
Interactivity v. Expectation

- H1 holds: more interactive the pattern, more likely user is to expect access
  - 106 times more expected for Click than Bg
  - 4 times more expected for Notification than Bg
  - Confidence intervals don’t overlap: Click > Bg
- Explicit authorization also shows significant increase
  - First Use: 2.2 times as likely
  - Launch: 1.9 times as likely
Likert Responses for First Use

Click much more expected, Bn too
H1: Interactivity v. Expectation

- The more interactive the pattern, the more likely user is to expect access
  - 106 times more expected for Click than Bg
  - 4 times more expected for Notification than Bg
  - Confidence intervals don’t overlap: Click > Bg
- Explicit authorization also shows significant increase
  - First Use: 2.2 times as likely
  - Launch: 1.9 times as likely
H2: Real World Freq vs. Expectation

- Location was generally most expected
  - Press, frequently seen in action bar, etc.?  
- No significant difference between mic and contacts
- Mic uses were very unexpected without interaction
H3: Effect of Prior Access

- More likely to expect $Bg$ access when prior event ($Bn$: 2.1, $Lch$: 1.7) indicated $Bg$ use could occur
- Prior event of $Click$ not significantly different from $Bg$
- *First Use* not significantly different from *Never* for second access
- *First Use* may condition users to expect a single access
  - Authorizing location on first use: lower $Bg$ expectation later!
Design Recommendations
Access Resources Interactively

• Camera, Mic, Media, Calendar: already interactive
  • Some legitimate outliers, but users should be made aware of them

• We recommend these uses always be interactive
  • E.g., enforce in market, audit outliers?
Interaction Grants Authorization

- No need for explicit authorization after interactions
  - “I just clicked ‘import contacts’! Why is it asking me again?”
- Minimize burden, more capacity for important decisions
- Ensure interactions are relevant
  - Perhaps use tool like AppTracer to help audit?
  - Access Control Gadgets  [Ringer et al. CCS-2016]
Separate Background Authorization

• Placing it near interactive use may fool users
  • We recommend placing it on start
• Future work: understand types of background use
  • Do users differentiate different background uses?
  • Does frequency, information flow, etc… matter?
Takeaways

- Balance awareness vs. invasiveness by integrating UI
- Android heading in right direction
  - But asks *too often* and *not enough*
- Most sensitive resources already used interactively
  - Aligns with user expectations
- Interactive / background authorization should be separated