What the App is That? Deception and Countermeasures in the Android User Interface

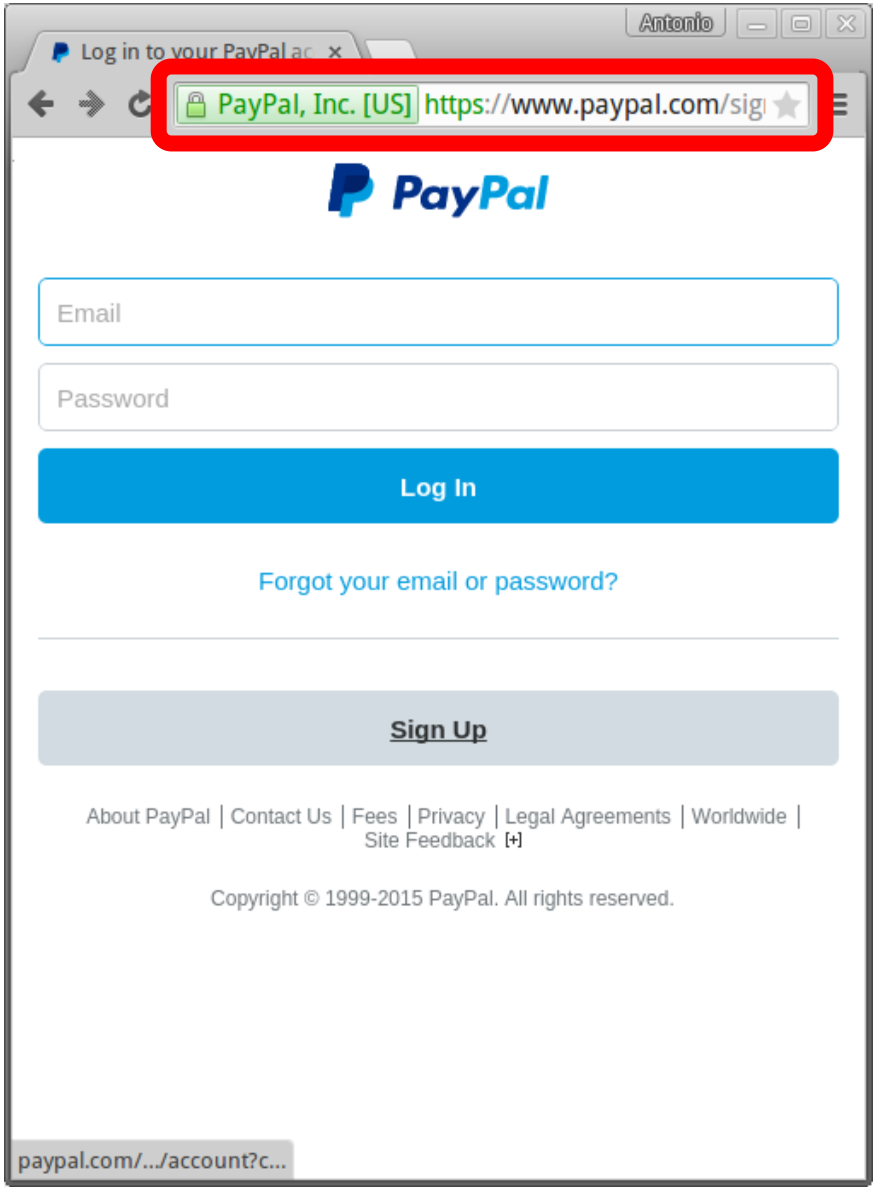
Antonio Bianchi	antoniob@cs.ucsb.edu
Jacopo Corbetta	jacopo@cs.ucsb.edu
Luca Invernizzi	invernizzi@cs.ucsb.edu
Yanick Fratantonio	yanick@cs.ucsb.edu
Christopher Kruegel	chris@cs.ucsb.edu
Giovanni Vigna	vigna@cs.ucsb.edu

University of California, Santa Barbara

THE COMPUTER SECURITY GROUP AT UC SANTA BARBARA

36th IEEE Symposium on Security and Privacy
Session: Android Security
May 20th, 2015

What am I interacting with?



Log in to your PayPal ac ×

Antonio

PayPal, Inc. [US] https://www.paypal.com/sig

PayPal

Email

Password

Log In

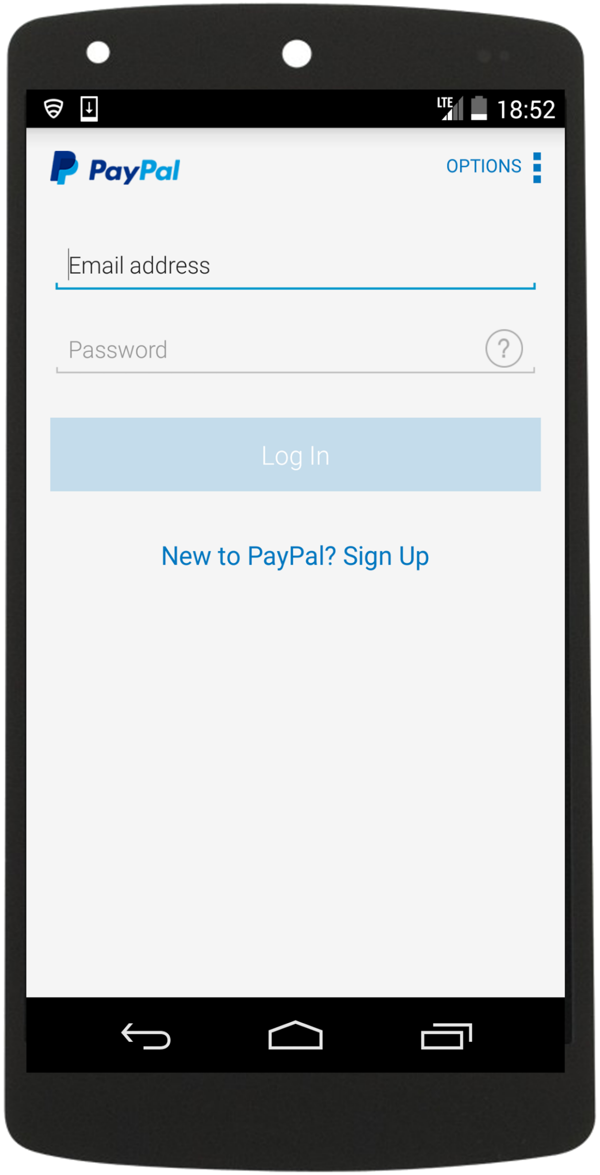
Forgot your email or password?

Sign Up

About PayPal | Contact Us | Fees | Privacy | Legal Agreements | Worldwide | Site Feedback [+]

Copyright © 1999-2015 PayPal. All rights reserved.

paypal.com/.../account?c...



18:52

PayPal

OPTIONS

Email address

Password

Log In

New to PayPal? Sign Up

Motivation

1) No origin indication

No information about the app a user is interacting with

2) No graphical separation

An app can "jump" on-top of another
An app can draw on-top of another

3) Incomplete compartmentalization

An app can know the app a user is currently interacting with

Motivation

<attack video>

Contributions

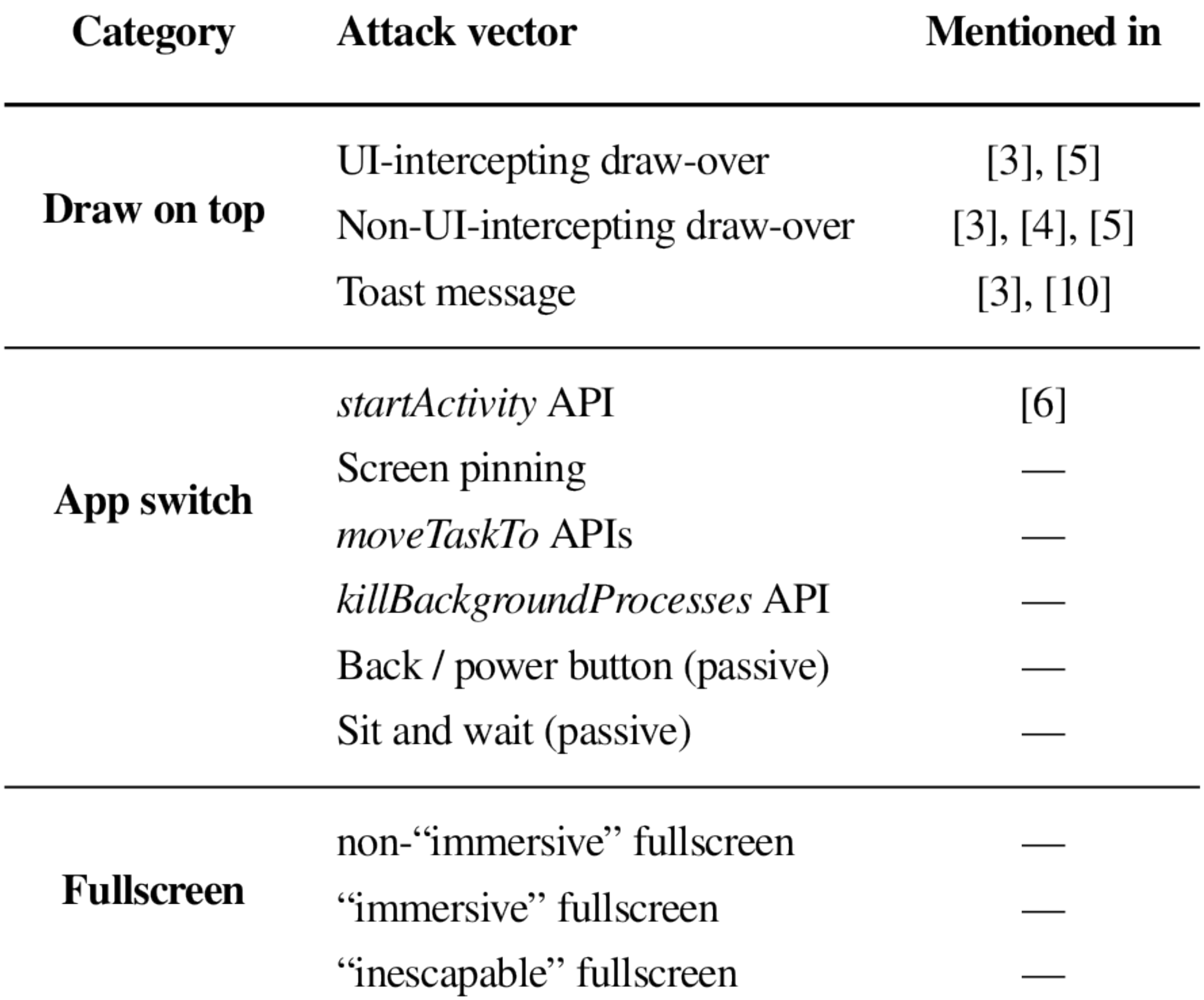
- **Systematic study**
  Study of the different techniques that can be used to perform “GUI-confusion” attacks in Android

- **Market-level defense**
  based on static analysis

- **On-device defense**
  based on UI modifications
  → evaluated with a user-study

Attacks

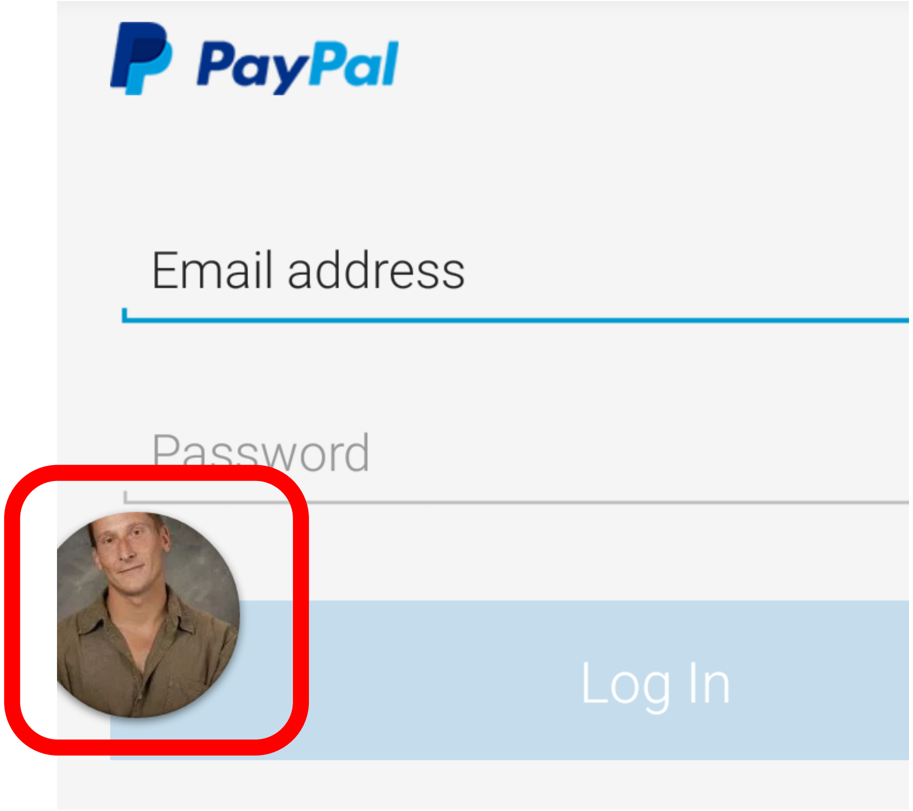
Exploiting missing graphical separation:



Category	Attack vector	Mentioned in
Draw on top	UI-intercepting draw-over	[3], [5]
	Non-UI-intercepting draw-over	[3], [4], [5]
	Toast message	[3], [10]
App switch	*startActivity* API	[6]
	Screen pinning	—
	moveTaskTo APIs	—
	killBackgroundProcesses API	—
	Back / power button (passive)	—
	Sit and wait (passive)	—
Fullscreen	non-“immersive” fullscreen	—
	“immersive” fullscreen	—
	“inescapable” fullscreen	—

Attacks

Exploiting missing graphical separation:



Attacks

Automatic state-exploration:

- automatic study of the complex Android API

- interesting finding:

  it is possible to create “inescapable” fullscreen windows

Attacks

Exploiting incomplete compartmentalization:

getting information about user interaction with other applications

- *getRunningTask* API (up to Android 4.4)
- */proc/<process_pid>/cgroups*
- */proc/<process_pid>/statm* [Chen 2014]

[Chen 2014] *Qi Alfred Chen, Zhiyun Qian, and Z. Morley Mao.*
"Peeking into Your App Without Actually Seeing it: Ui State Inference and Novel Android Attacks."
USENIX Security 2014

Defense

- Automatic at Market-level
  - Using static analysis to automatically identify applications that can potentially perform "GUI-confusion" attacks

Defense — Market-level

An app is classified as potentially malicious iff:

1) It uses a technique to detect which app the user is interacting with.
2) It uses a technique to jump/draw on-top of other apps.

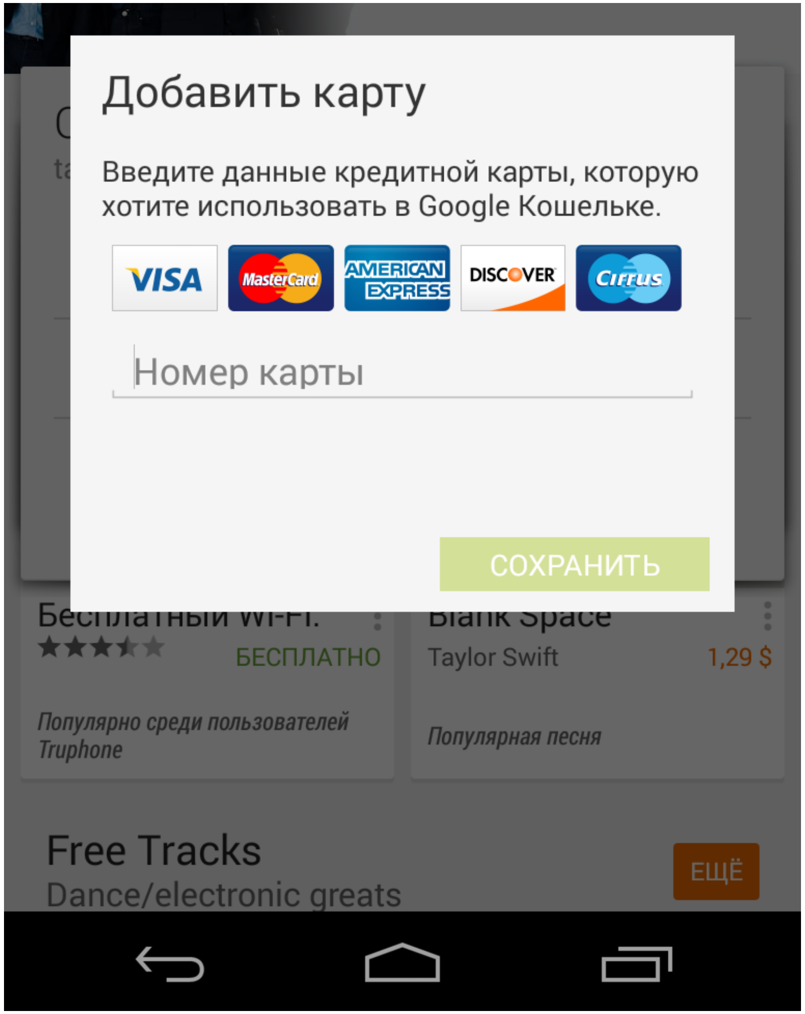
We use code slicing techniques to detect called APIs and their parameters

3) There is a connection between code locations where 1) and 2) happen.

Control flow analysis

Defense — Market-level

A detected sample (from the *svpeng* malware family)



Добавить карту

Введите данные кредитной карты, которую хотите использовать в Google Кошельке.

Номер карты

Defense — Market-level

Dataset	Detected
500 randomly selected apps	2: “app-lockers”
500 “top free” apps on Google Play	2: “app-lockers” 21: interfering with UI (e.g., showing disruptive ads) 3: false positives
1,260 apps from the “Android Malware Genome” project	21: samples from the *DroidKungFu* malware family, aggressively displaying an Activity on top of any other 4: false positives

Defense — Market-level

Dataset	Detected
500 randomly selected apps	2: “app-lockers”
500 “top free” apps on Google Play	2: “app-lockers” 21: interfering with UI (e.g., showing disruptive ads) 3: false positives
1,260 apps from the “Android Malware Genome” project	21: samples from the *DroidKungFu* malware family, aggressively displaying an Activity on top of any other 4: false positives

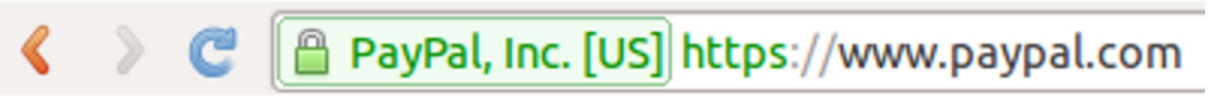
Defense — Market-level

Dataset	Detected
500 randomly selected apps	2: “app-lockers”
500 “top free” apps on Google Play	2: “app-lockers” 21: interfering with UI (e.g., showing disruptive ads) 3: false positives
1,260 apps from the “Android Malware Genome” project	21: samples from the *DroidKungFu* malware family, aggressively displaying an Activity on top of any other 4: false positives

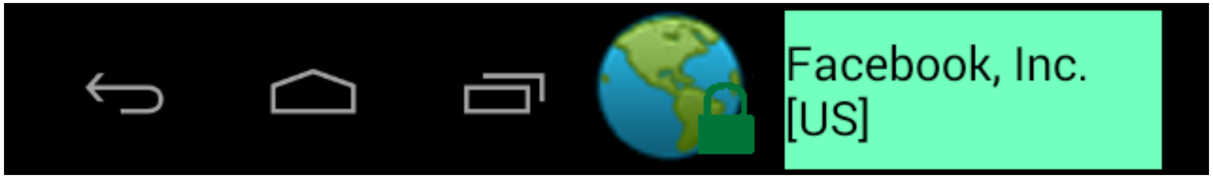
Defense — On-device

Modifications to the Android graphical user interface

Allow users to reliably know what they are interacting with



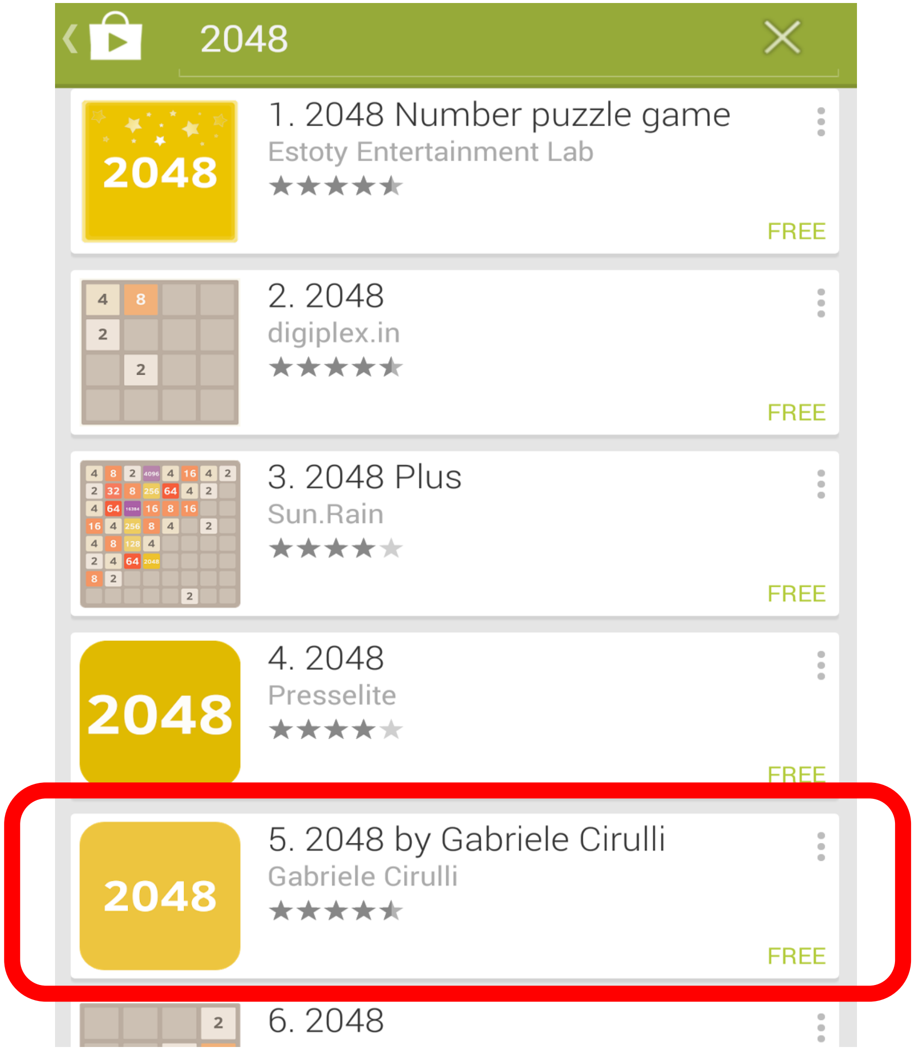
PayPal, Inc. [US] https://www.paypal.com



Facebook, Inc. [US]

Defense — On-device

- Understanding the "origin" of an app
  - We cannot trust the Market!



2048

1. 2048 Number puzzle game
Estoty Entertainment Lab
FREE

2. 2048
digiplex.in
FREE

3. 2048 Plus
Sun.Rain
FREE

4. 2048
Presselite
FREE

5. 2048 by Gabriele Cirulli
Gabriele Cirulli
FREE

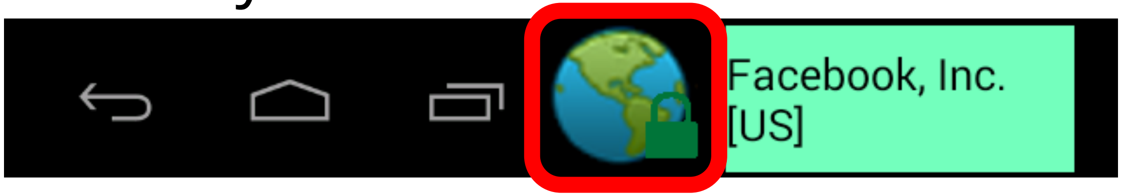
6. 2048

Defense — On-device

- Understanding the “origin” of an app
  - We rely on the already-existing SSL Extended Validation (EV) infrastructure to validate the *author* of an app.
  - An app must specify a domain *D* (controlled by the app’s developer)
    - *D* must contain a file with the public key used to sign the app.
    - *D* needs to be certified using an SSL EV certificate
  - We show the “organization name” from the EV certificate of *D.*

Defense — On-device

- Showing the security indicator in an unobtrusive but reliable way.
  - We use the “navigation bar” to show a security indicator
  - We use a “secret image”
    - only known to the user and the operating system (selected by the user during device first-boot)
    - avoid malicious “fullscreen” apps spoofing the security indicator


Facebook, Inc.
[US]

Defense - Dynamic

<defense video>

Evaluation

- User study
  - Evaluating the effectiveness of our on-device defense
  - Subjects recruited on Mechanical Turk
  - Subjects interact with an emulated device using their browser

Evaluation

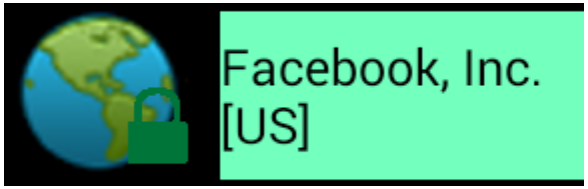
- 308 subjects divided in 3 groups
  - G1
    - stock Android system (no on-device defense)
  - G2
    - on-device defense in place
    - no additional training
  - G3
    - on-device defense in place
    - subjects aware of the possibility of attacks
    - subjects received additional training about security-indicator functionality

Evaluation

- Subjects are asked to interact with the Facebook app multiple times

- After each interaction, subjects are asked if they think they have interacted with the original Facebook app

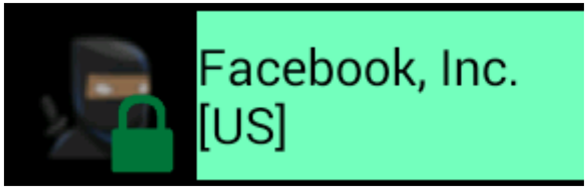
- 4 interactions (in randomized order) are evaluated
  - B_1 and B_2
    - the subject is not attacked



Facebook, Inc.
[US]

  - A_{std}
    - the malicious app covers the legitimate one

  - A_{full}
    - the malicious app also shows a spoofed security indicator (by using a fullscreen Window)



Facebook, Inc.
[US]

Evaluation

Answering correctly during:	G1 stock Android	G2 on-device defense unaware of attacks no additional training	G3 on-device defense aware of attacks additional training
A_{std}	19.19%	64.52%	68.97%
A_{full}	17.17%	76.34%	74.14%
all 4 interactions	2.02%	53.76%	56.90%

Evaluation

Answering correctly during:	G1 stock Android	G2 on-device defense unaware of attacks no additional training	G3 on-device defense aware of attacks additional training
A_{std}	19.19%	64.52%	68.97%
A_{full}	17.17%	76.34%	74.14%
all 4 interactions	2.02%	53.76%	56.90%

Evaluation

Answering correctly during:	G1 stock Android	G2 on-device defense unaware of attacks no additional training	G3 on-device defense aware of attacks additional training
A_{std}	19.19%	64.52%	68.97%
A_{full}	17.17%	76.34%	74.14%
all 4 interactions	2.02%	53.76%	56.90%

Conclusions

- We studied the problem of “GUI-confusion” attacks in Android

- We propose:
  - a market-level defense, based on static analysis
  - an on-device defense based on UI modifications
    → evaluated with a user study

- Source code of the on-device defense:
  https://github.com/ucsb-seclab/android_ui_deception

“That’s all folks!”

Questions?