Betrayed by the Android UI

Yanick Fratantonio
EURECOM
@reyammer

Insomni'hack 2019
March 22nd, 2019

Whoami

- Yanick Fratantonio (@reyammer)
- Academic
  - Assistant Professor at Eurecom (Nice area, France)
  - PhD @ UC Santa Barbara
- Research focus: Mobile Systems Security & Privacy
- Hacking / Capture The Flag teams
  - Shellphish (UCSB), NOPS (Eurecom)
  - The Order of the Overflow (DEFCON CTF organizers) ← bad idea

MOBISEC class

- Recently released all material for my Mobile Security class
- https://mobisec.reyammer.io
- Material
  - 800+ slides on the topic
  - Coming next week: wargame site dedicated to mobile security
    - Reversing challenges / exploitation challenges

Today's talk: Android UI Security

- UI security matters

- UI attacks are real
  - They exist
  - They are practical
  - They are difficult to eradicate (some of them: open research problem)

Primer on Android security

- Users can install third-party apps
- Third-party apps are, by default, sandboxed
  - Apps have private storage
  - Their capabilities are monitored via the permission system
  - Interaction only through well-defined IPC mechanisms
- Very different than usual PC / laptops!
  - If an attacker gets code execution on my laptop, it's game over
  - Not the case for my Android phone!

Many low-level security mechanisms

- Many efforts to
  - reduce attack surface
  - tighter adherence to principle of least privilege
  - exploit mitigation techniques
  - permission system refinement
  - new permission policies (e.g., clipboard access only for foreground apps)
  - SELinux policies / contexts
  - limited access to /proc & co.
- Great talk by Nick Kralevich @ BHUSA'17 on Google's work to shrink the attack surface

UI security matters

- UI attacks can bypass many low-level mechanisms
- Android's Achilles' heel
  - Apps have full control of your screen
  - Apps can do UI "tricks"
  - Not well understood
- Lack of Trusted UI prevents using mobile devices to control security-critical systems, medical devices, E-IDs, ...

UI attacks

- What is a UI attack?
  - An attack involving UI that somehow affects the CIA triad
  - User deception

- Focus on "imperceptible attacks"
  - Even a security expert cannot notice an attack is going on...
  - ... even if I tell you that you are under attack!

- Example of non-imperceptible attack
  - Web phishing: a user can always check "the ~~green~~ lock" + domain name

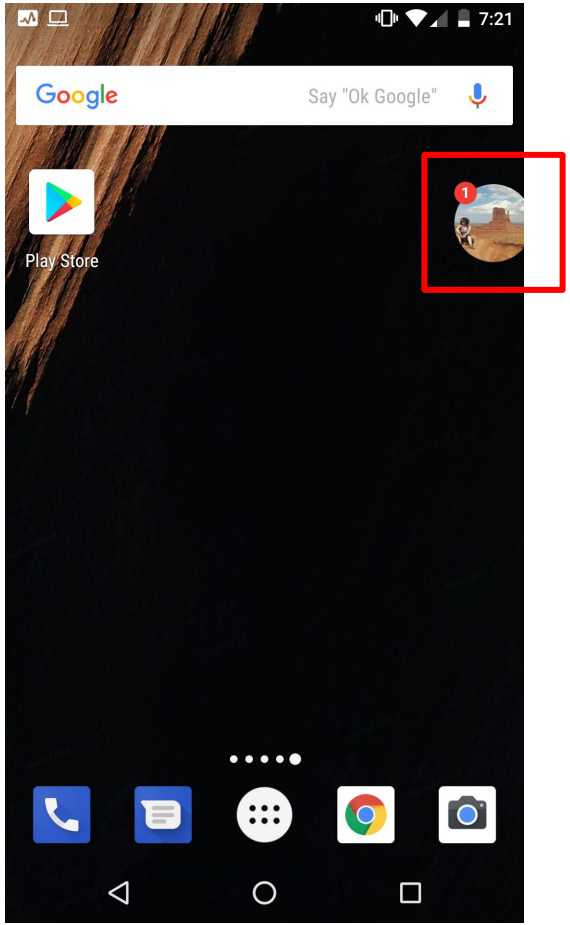
Two big classes

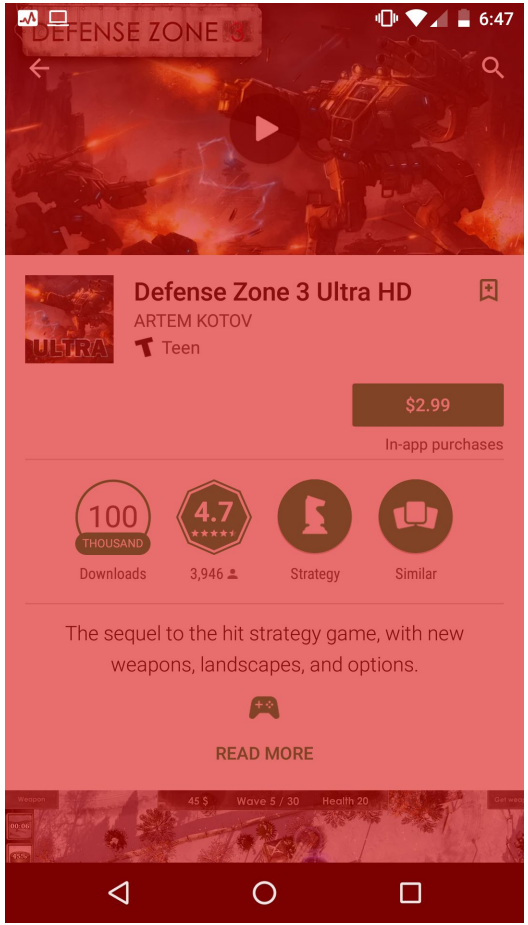
- Clickjacking
  - Attacker lures the user to “click” somewhere
  - Usual goal: privilege escalation / confused deputy
- Phishing
  - Attacker lures the user to insert her credentials / private data somewhere (and leak them to the attacker)
- But there are some other twists & tricks to abuse password managers, instant apps, ...

Clickjacking 101

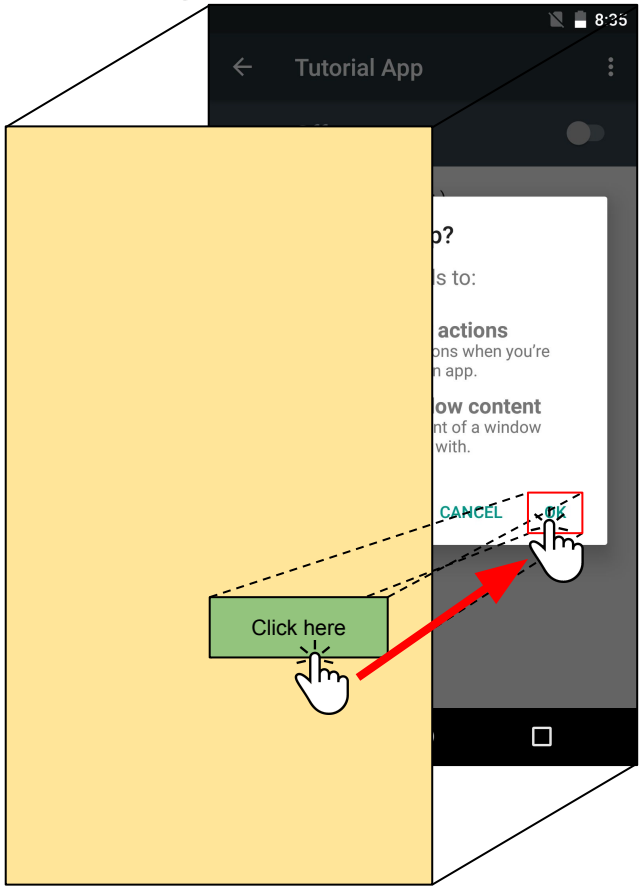
“Draw on top” permission

- Draw arbitrary windows/overlays on top of the screen
  - Can be completely custom: shape, content, transparency, position
  - Can be clickable ⊕ passthrough
- This permission is used quite often
  - 454 out of 4,455 top apps (10.2%)
  - Used by Facebook, Skype, Uber, LastPass, ...
- Automatically granted to apps from the Play Store*
  - *NOTE: it is possible that this will change soon -- I've heard rumors ;-)


7:21
Google
Say "Ok Google"
Play Store
1


6:47
Defense Zone 3 Ultra HD
ARTEM KOTOV
Teen
$2.99
In-app purchases
100
THOUSAND
Downloads
4.7
3,946
Strategy
Similar
The sequel to the hit strategy game, with new weapons, landscapes, and options.
READ MORE

Traditional Clickjacking



UI Redressing Attacks on Android Devices Revisited
Niemietz & Schwenk
BH ASIA 2014

Multi-step clickjacking (?)

- Multi-step clickjacking: some attacks require 2+ clicks
- Challenges
  - When to transition to the next stage?
  - What if the user clicks “somewhere else”?
  - Tricky because the first click lands, by definition, on the *victim* app
    - The malicious app is not notified about clicks landing elsewhere
    - Exception: FLAG_WATCH_OUTSIDE_TOUCH flag, but the click’s coordinates are set to (0,0) if click lands on another app
      - Where did the user clicked?
      - Wheeereeeeee?

Attack: Context-aware Clickjacking

- So, the attacker does not know the coords of clicks landing outside its malicious app...

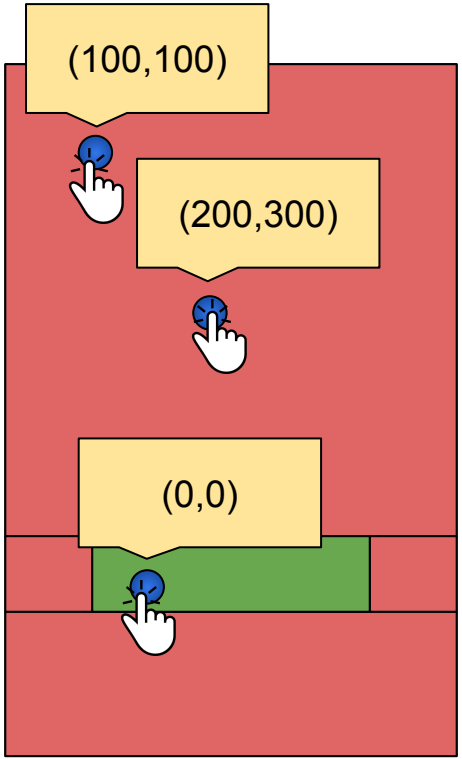
- But what if there is only “one way” for a click to not reach the malicious app?

Multi-step Clickjacking

Clicks do **NOT** go through

Clicks go through

- We know the user clicked on the “target” button
- We know we need to transition to the next step

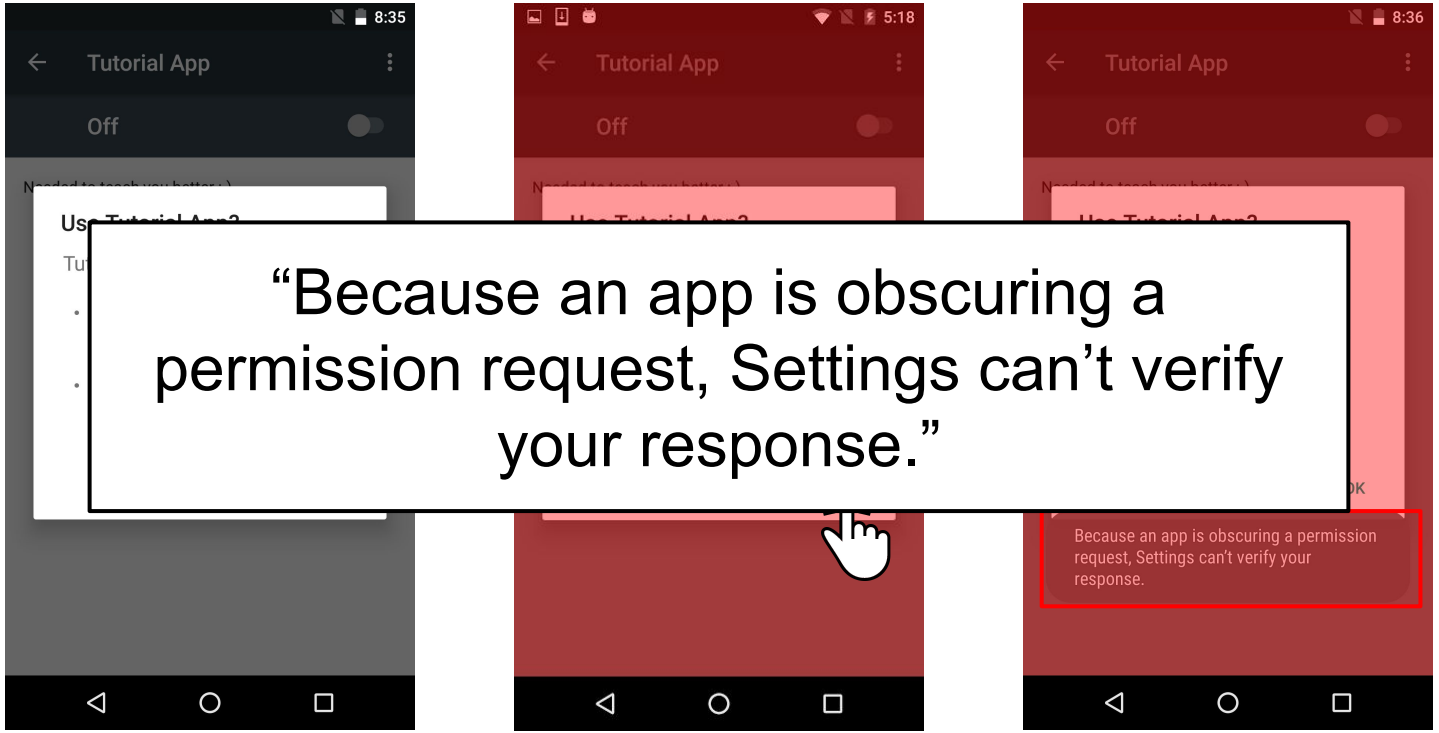


Cloak & Dagger
IEEE S&P’17

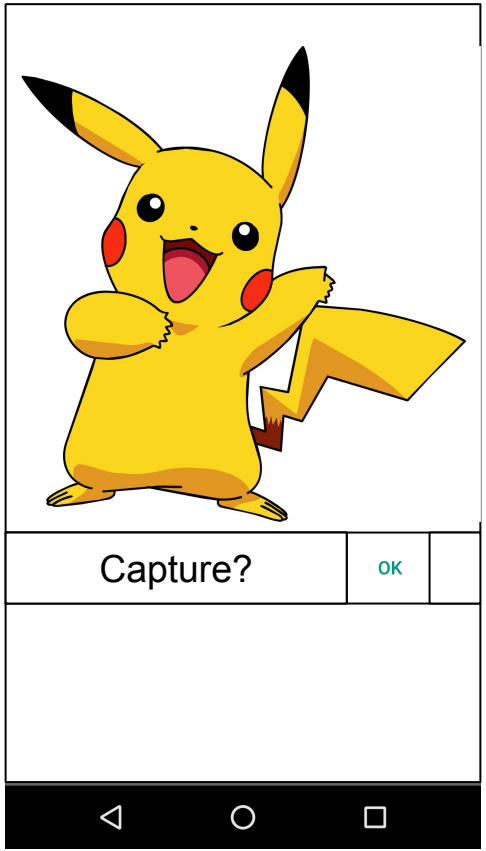
Protection against clickjacking

- Clickjacking attack is old
- Google introduced the “obscured” flag
  - When the user clicks on a widget, FLAG_WINDOW_IS_OBSCURED is set if “an overlay was covering the receiving widget”
  - An app can decide to “not trust” the click
- Another option: `setFilterTouchesWhenObscured()`

Obscured Flag Defense Mechanism



Obscured Flag Bypass



Context-Hiding Attack

Cloak & Dagger
IEEE S&P'17

From “Draw on top” to a11y

- Android Accessibility Service (a11y)
  - In theory: mechanism for apps to assist users with disabilities
  - In practice: super powerful mechanism abused by benign/malicious apps

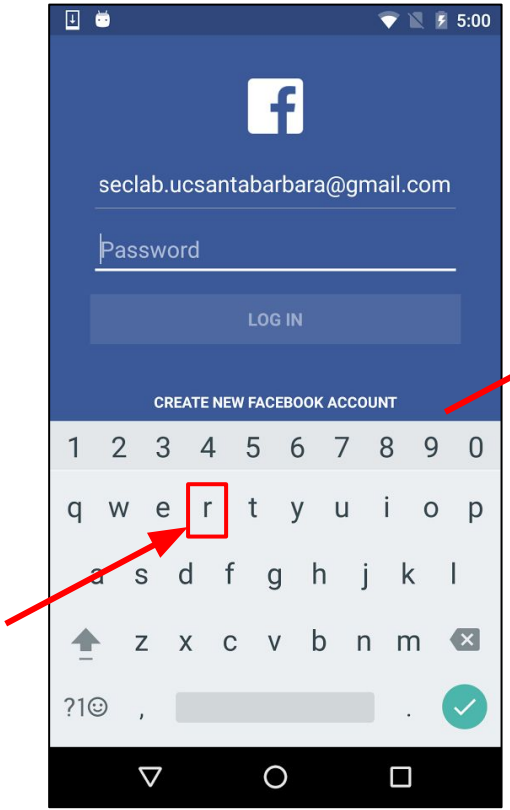
- “Features”
  - App is notified for each UI event
  - App can inject UI events (e.g., clicks)

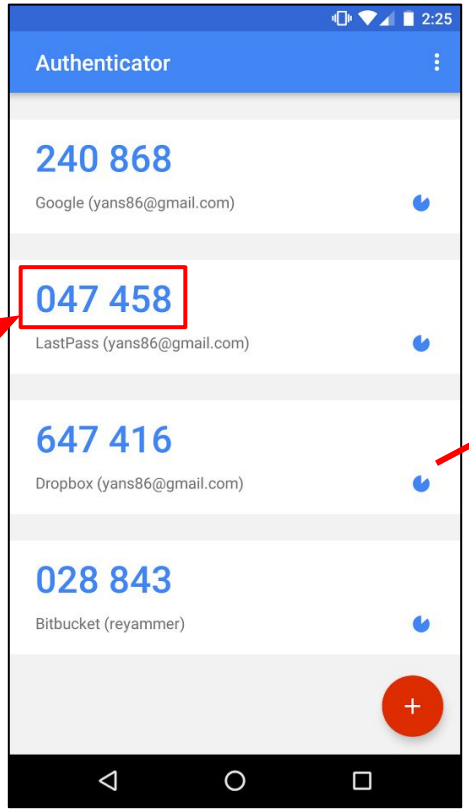
A11y security

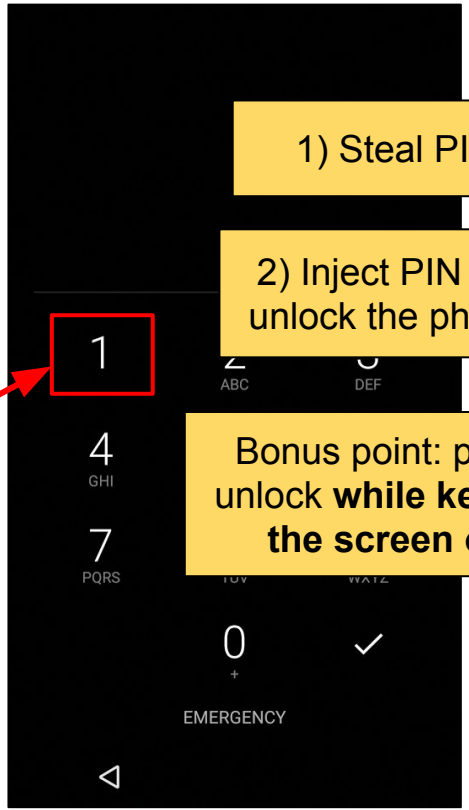
- Very powerful, but not God-mode (in theory...)

> *"Since an event contains the text of its source* ***privacy can be compromised by leaking sensitive information*** *such as passwords. To address this issue* ***any event fired in response to manipulation of a PASSWORD field does NOT CONTAIN the text of the password.***"

Attack: a11y on steroids







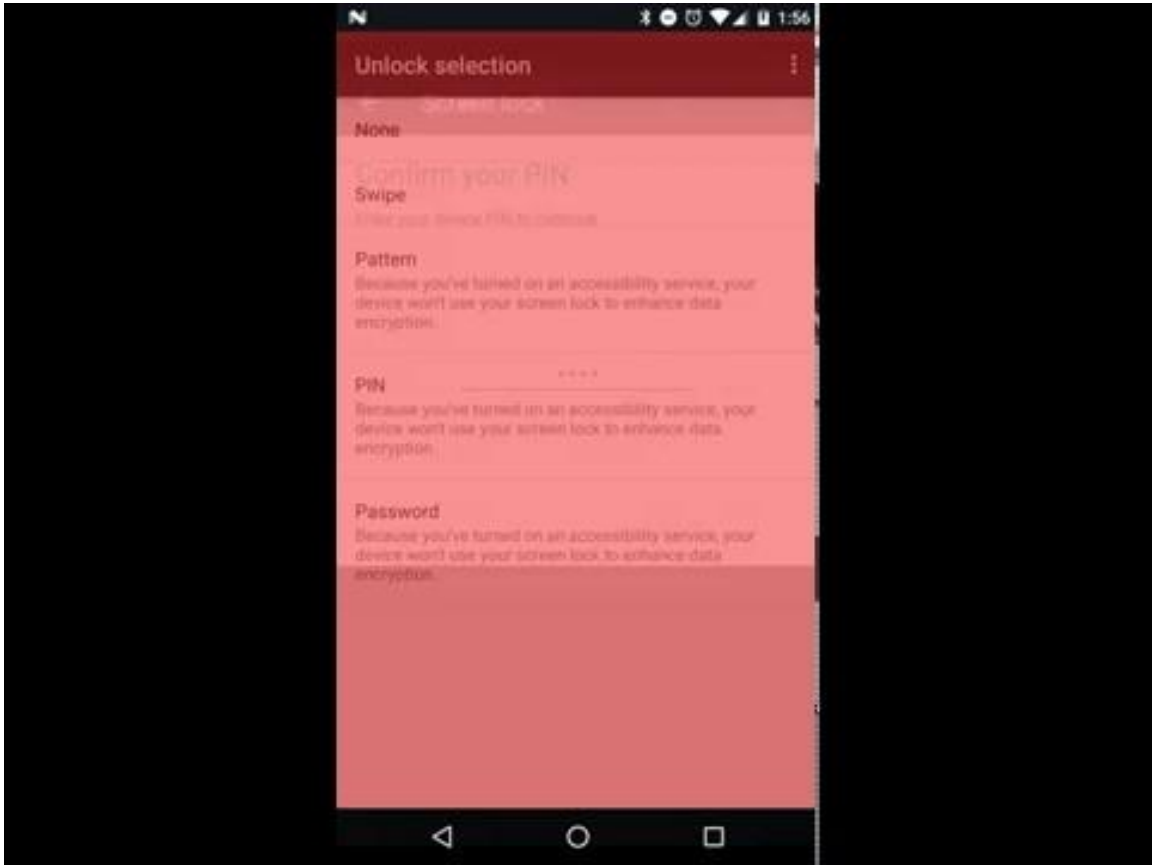
1) Steal PIN

2) Inject PIN and unlock the phone!

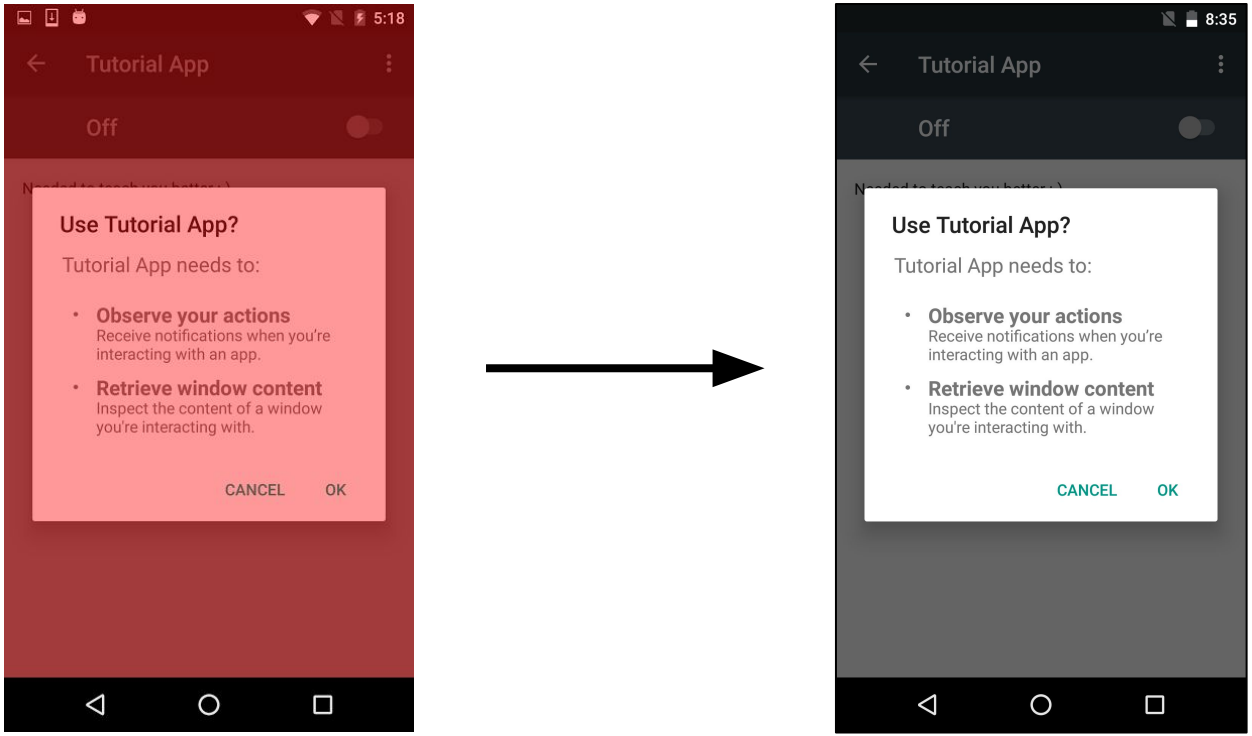
Bonus point: phone unlock **while keeping the screen off!**


5:49
In this tutorial, humans will be represented by green droids, like the one below
Off
NEXT

Ransomware Example



“Hide overlays” defense



“Hide overlays” defense

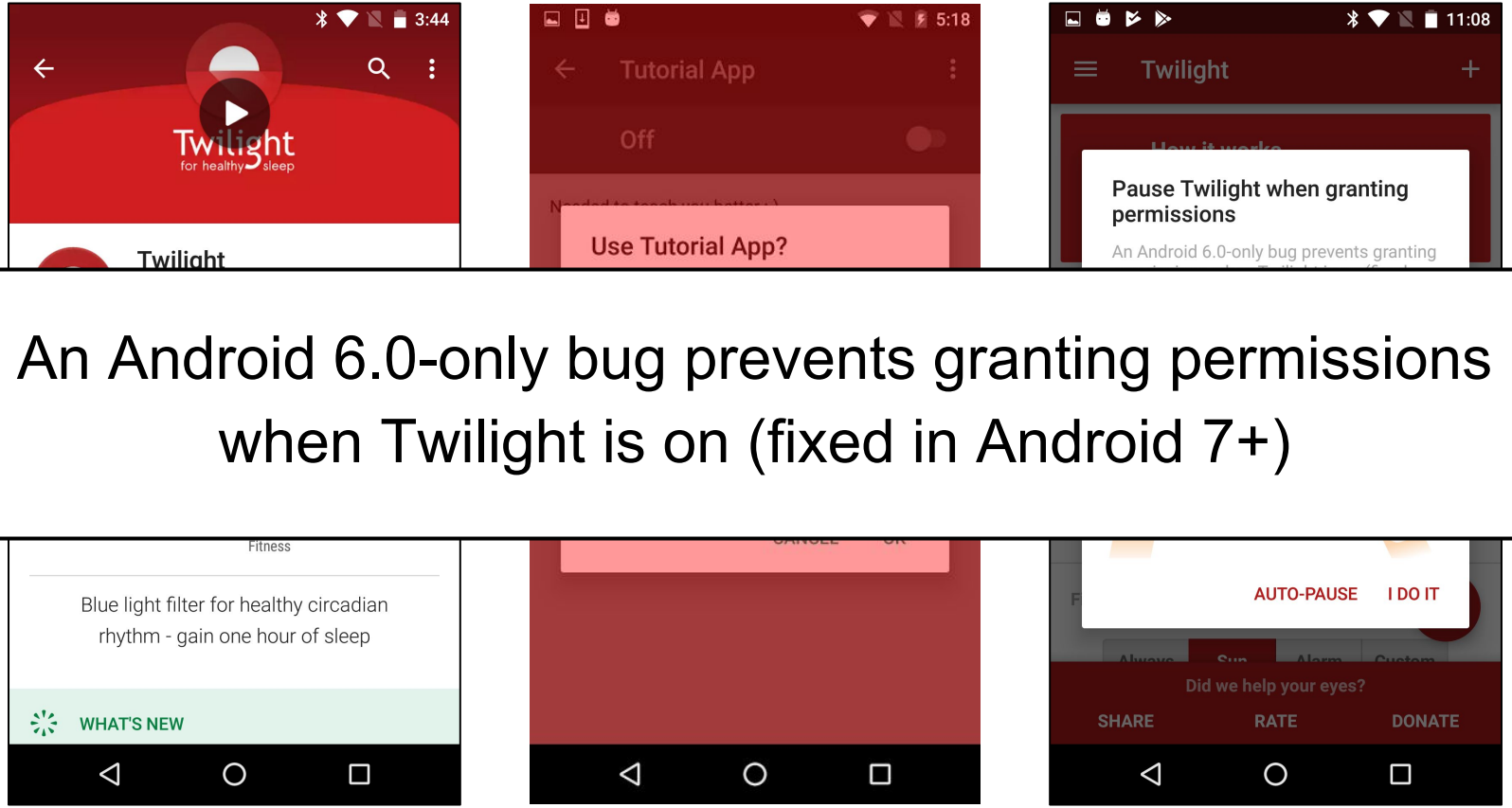
- It works!

- I believe it is enough to prevent clickjacking

But...

- …are these defenses widely deployed?
  - Not really: only system apps can use "hide overlays" trick
- What about the well-known obscured flag? Is it used?
- “A friend told me…”

Twilight



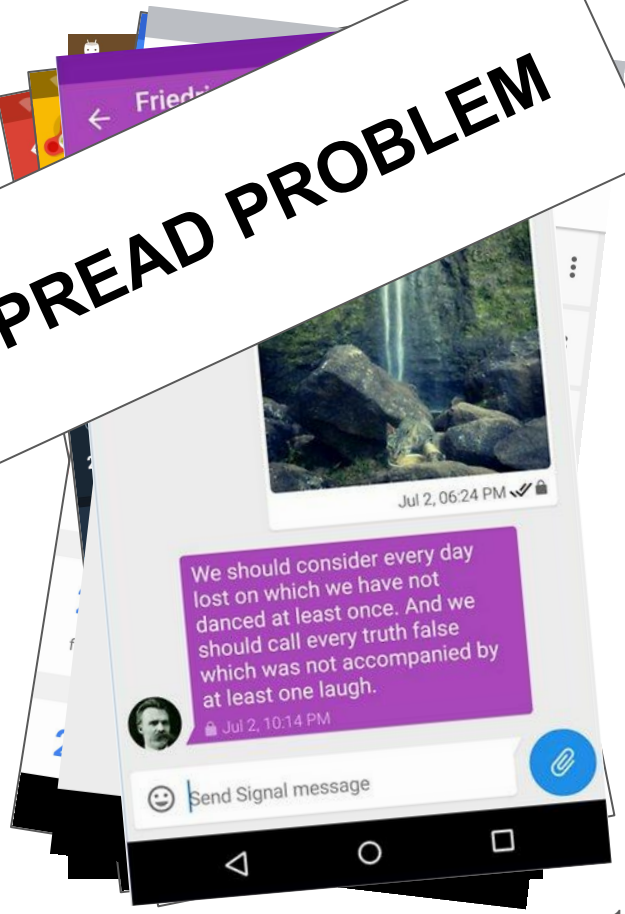
An Android 6.0-only bug prevents granting permissions when Twilight is on (fixed in Android 7+)

Clickjacking vulns are still widespread

Many more targets

- Google Play Store
- GMail
- Google Authenticator
- Twitter, Facebook
- Google Driv
- Signal
-
-
-

CLICKJACKING IS STILL A WIDESPREAD PROBLEM



Disclosure & Reaction

- **Twitter**: “After further review, we do not plan to addre
  to the UX issues you mention”

- **Signal**: “...without an effe
  for us to do here.”

BACKWARD COMPATIBILITY

...require
..., and our threat
...eration.”

ClickShield: Are You Hiding Something?
Towards Eradicating Clickjacking on Android

Simon Pak Ho Chung
Georgia Institute of Technology, USA
pchung34@mail.gatech.edu

Andrea Lanzi
Universita' degli Studi di Milano, Italy
andrea.lanzi@unimi.it

Yanick Fratantonio
EURECOM, France
yanick.fratantonio@eurecom.fr

Andrea Possemato
EURECOM, France
andrea.possemato@gmail.com

Wenke Lee
Georgia Institute of Technology, USA
wenke.lee@gmail.com

1 INTRODUCTION

Mobile devices are widespread and the
significant corpus of research. One
offensive research, which focuses
highlight vulnerabilities. Within
works has specifically focus
Many of these works have
ing attacks [4, 6, 9, 14,
malicious app into
words) into a wi
issue enabling
they are in
malicio
A
in

user-interface (UI) attacks, the com-
solved problem. On the contrary,
still an open problem for mo-
and industry solutions are
real-world for backward
, as a consequence,
Store remain, to
acks.
terface, this
ce for an
s. We
ew

UI security previously not really understood / taken seriously

Back to the “obscured flag”...

- Not only it can be easily bypassed...

Back to the “obscured flag”...

- Not only it can be easily bypassed...
- ... but #1: misleading documentation

FLAG_WINDOW_IS_OBSCURED docs

> *“This flag indicates that the window that received this motion event is **partly** or wholly obscured by another visible window above it.”*

- This is not the case: if the click does not go through other overlays, the obscured flag does not kick in
- Google knows about it...

FLAG_WINDOW_IS_PARTIALLY_OBSCURED

```
/**
 * This flag indicates that the window t                          partly
 * or wholly obscured by anoth                          lag is set to true
 * even if the event                          bscured area.
 * A securi                    this flag to identify situations in which
                    covered up part of its content for the purpose
                    or hijacking touches.  An appropriate response might be
               suspect touches or to take additional precautions to confirm the user's
     ual intent.
 *
 * Unlike
 * @hide
 */
public st
```

Same as FLAG_WINDOW_IS_OBSCURED

"Unlike FLAG_WINDOW_IS_OBSCURED, this is actually true.

Back to the “obscured flag”...

- Not only it can be easily bypassed...
- ... but #1: misleading documentation

Back to the “obscured flag”...

- Not only it can be easily bypassed...
- ... but #1: misleading documentation
- ... but #2: it could be abused to mount even worse attacks!

Attack: Invisible Grid Attack

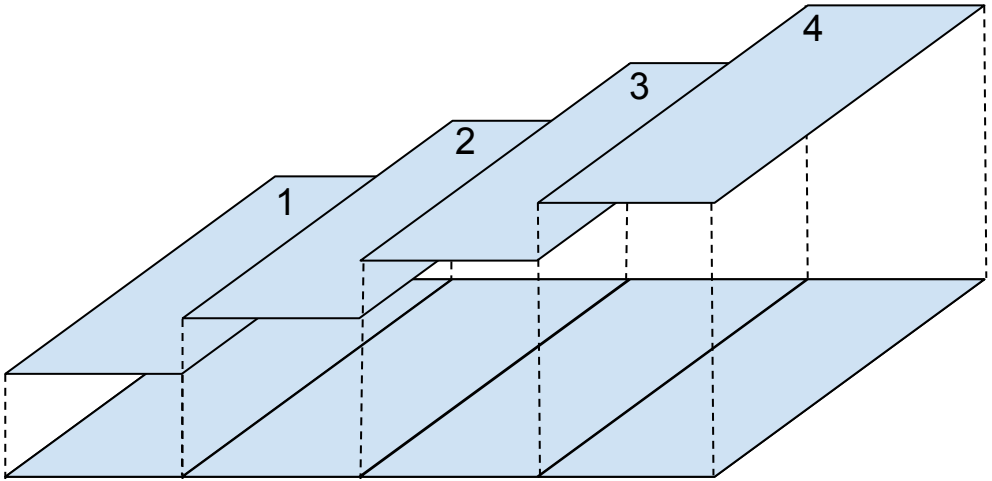
- This attack can record all “keystrokes”
  - It only relies on the “draw on top” permission

Attack: Invisible Grid Attack

- This attack can record all “keystrokes”
  - It only relies on the “draw on top” permission

- It abuses the “obscured flag” security mechanism

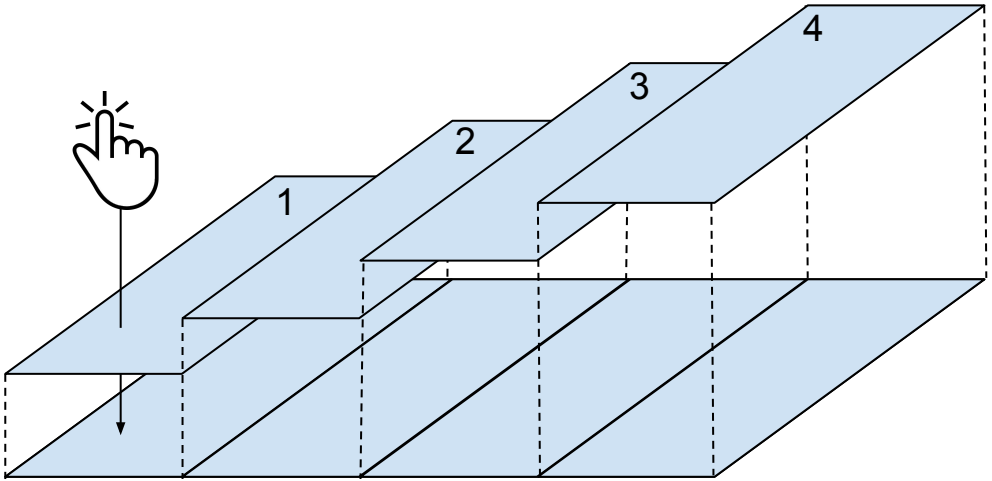
Attack: Invisible Grid Attack



Overlays are drawn

- Invisible

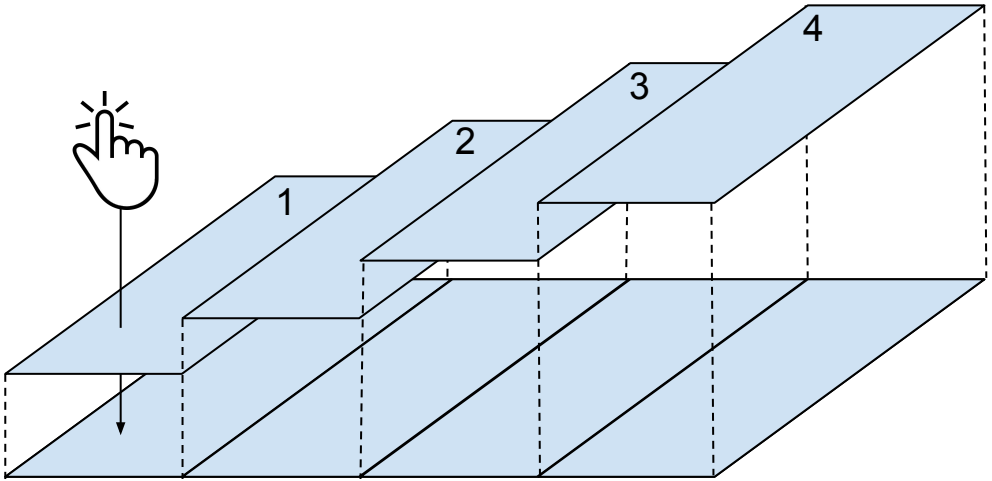
Attack: Invisible Grid Attack



Overlays are drawn
- Invisible
- Clicks passthrough

Attack: Invisible Grid Attack

Where did the user click?



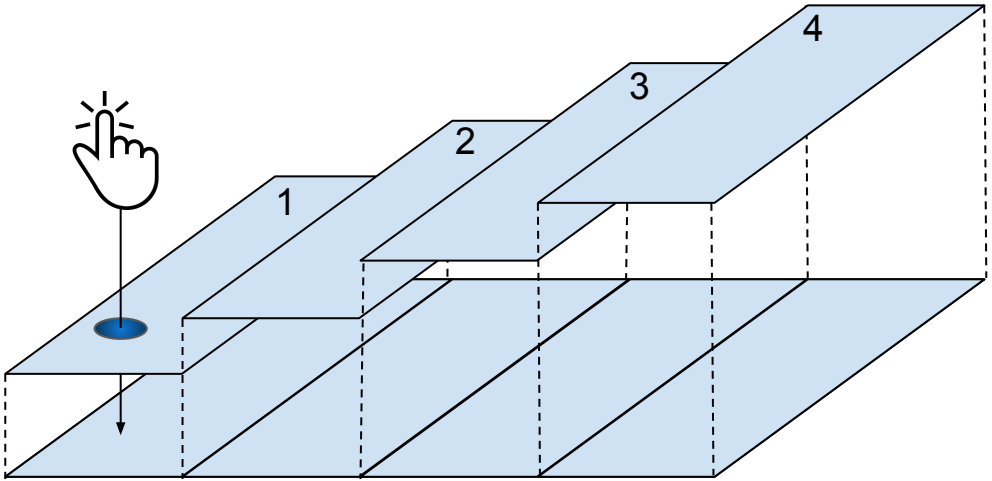
Overlays are drawn
- Invisible
- Clicks passthrough
- FLAG_WATCH_OUTSIDE_TOUCH

Overlay #

1	MotionEvent
2	MotionEvent
3	MotionEvent
4	MotionEvent

Attack: Invisible Grid Attack

Where did the user click?



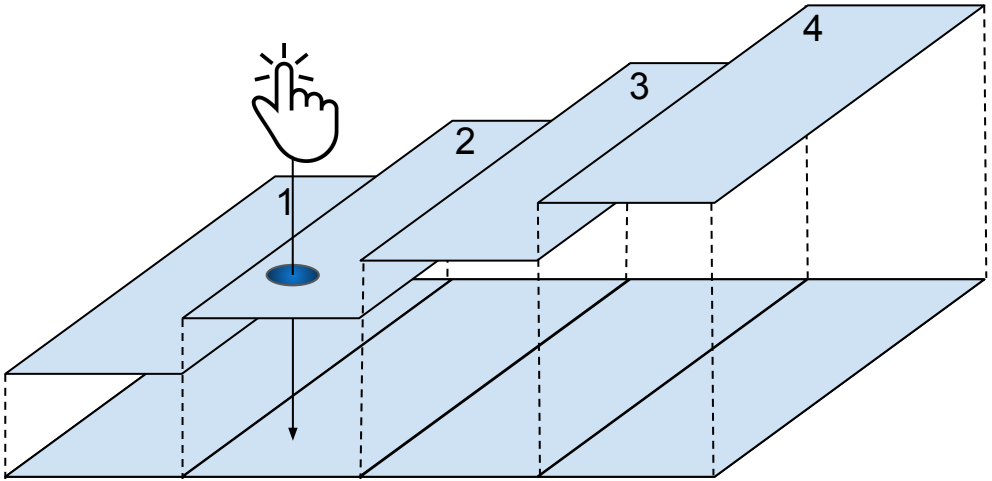
Overlays are drawn
- Invisible
- Clicks passthrough
- FLAG_WATCH_OUTSIDE_TOUCH

The “obscured” flag is set accordingly!

Overlay #		
1	MotionEvent	Not obscured
2	MotionEvent	Not obscured
3	MotionEvent	Not obscured
4	MotionEvent	Not obscured

Attack: Invisible Grid Attack

Where did the user click?

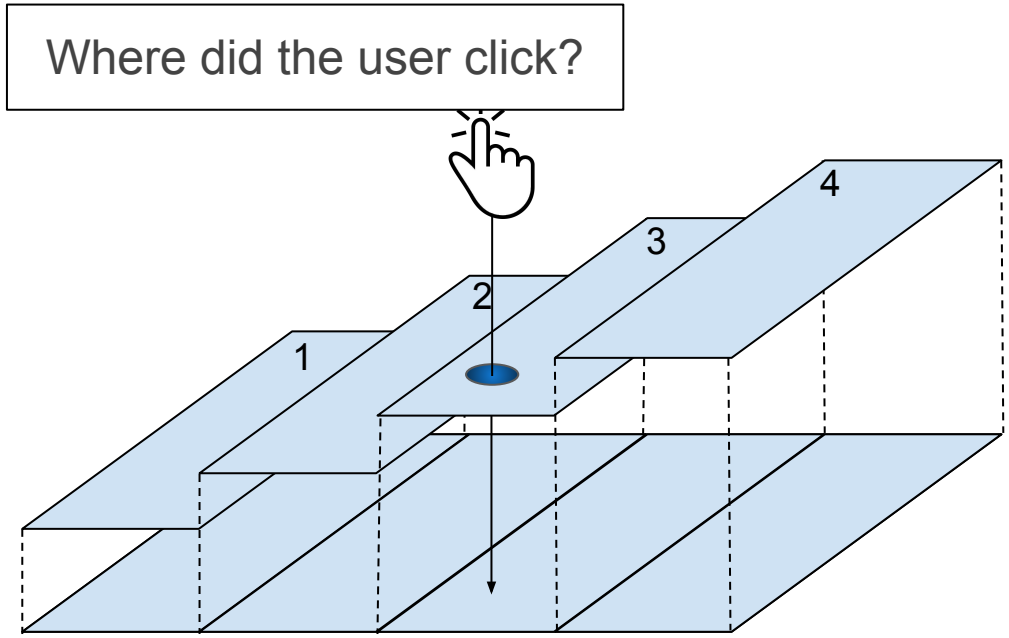


Overlays are drawn
- Invisible
- Clicks passthrough
- FLAG_WATCH_OUTSIDE_TOUCH

The “obscured” flag is set accordingly!

Overlay #		
1	MotionEvent	Obscured
2	MotionEvent	Not obscured
3	MotionEvent	Not obscured
4	MotionEvent	Not obscured

Attack: Invisible Grid Attack

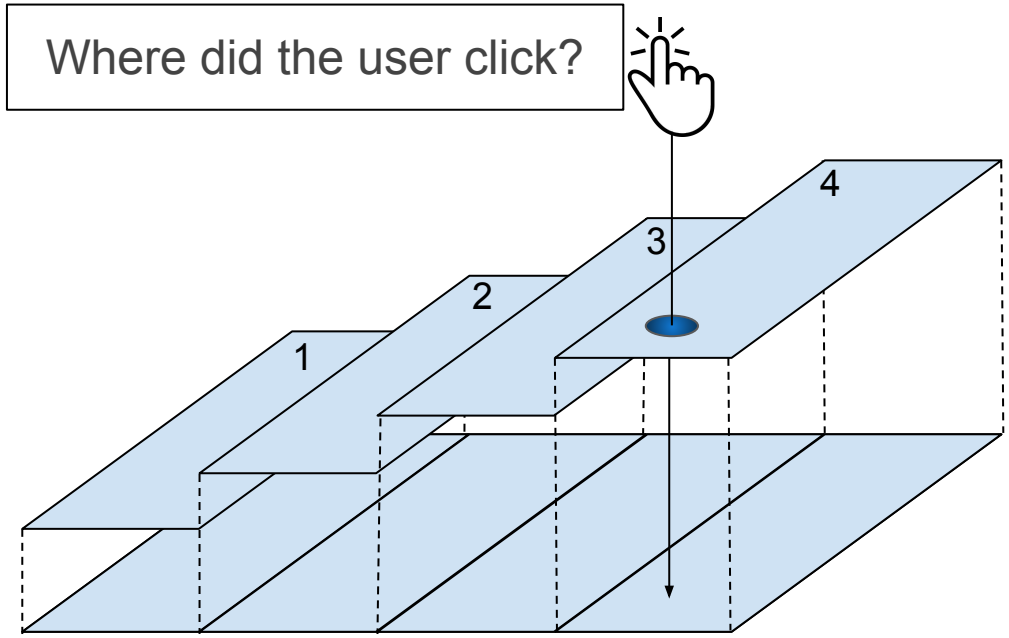


Overlays are drawn
- Invisible
- Clicks passthrough
- FLAG_WATCH_OUTSIDE_TOUCH

The “obscured” flag is set accordingly!

Overlay #		
1	MotionEvent	Obscured
2	MotionEvent	Obscured
3	MotionEvent	Not obscured
4	MotionEvent	Not obscured

Attack: Invisible Grid Attack

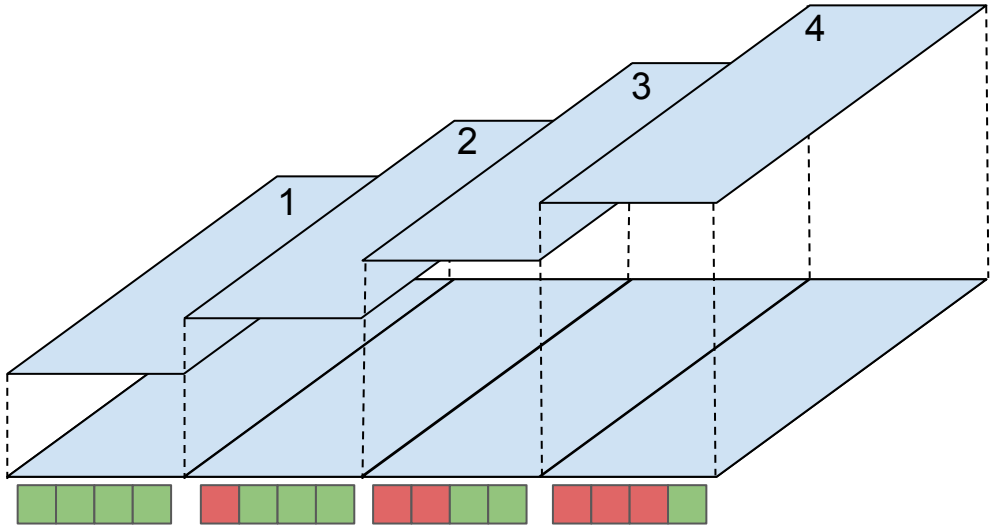


Overlays are drawn
- Invisible
- Clicks passthrough
- FLAG_WATCH_OUTSIDE_TOUCH

The “obscured” flag is set accordingly!

Overlay #		
1	MotionEvent	Obscured
2	MotionEvent	Obscured
3	MotionEvent	Obscured
4	MotionEvent	Not obscured

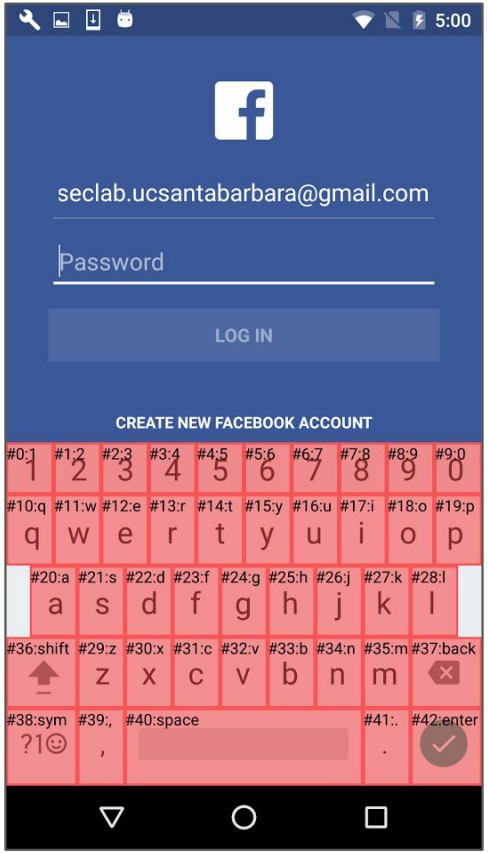
Attack: Invisible Grid Attack



Security mechanism used as side-channel!

The attacker can use these patterns to infer where the user clicked!

Attack: Invisible Grid Attack



These overlays are drawn invisible during a real attack

Disclosure of “a11y on steroids”

- Bug marked as “Won’t fix, work as intended” (September 30th)
- Bug marked as “High severity” (October 18th)

- Downgraded to “Won’t fix” because “limiting those services would render the device unusable” (November 28th)
- “We will update the documentation” (May 4th)
- AND THEY DID!!!11!1!

a11y documentation “patch”

- AccessibilityEvent’s “security note” is silently removed
  - June 6th version vs current version

a11y documentation “patch”

- AccessibilityEvent’s “security note” is silently removed
  - June 6th version vs current version

- “Patch the documentation, not the code”

a11y documentation “patch”

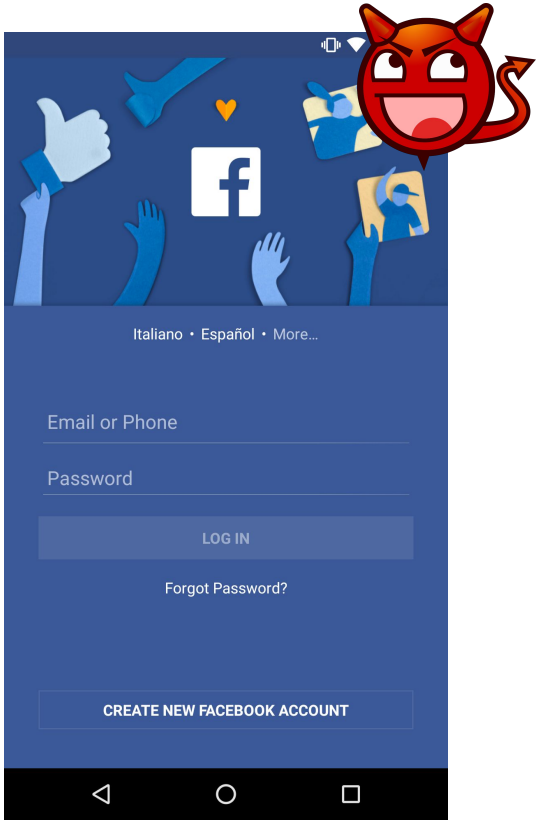
- AccessibilityEvent’s “security note” is silently removed
  - June 6th version vs current version

- “Patch the documentation, not the code”

- Found a 0day in the docs, still waiting for CVE ;-)

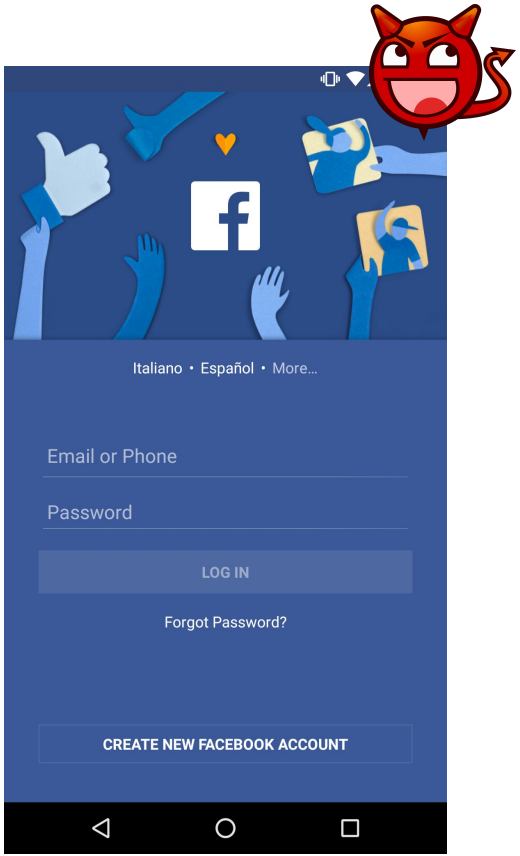
Mobile Phishing

The key problem



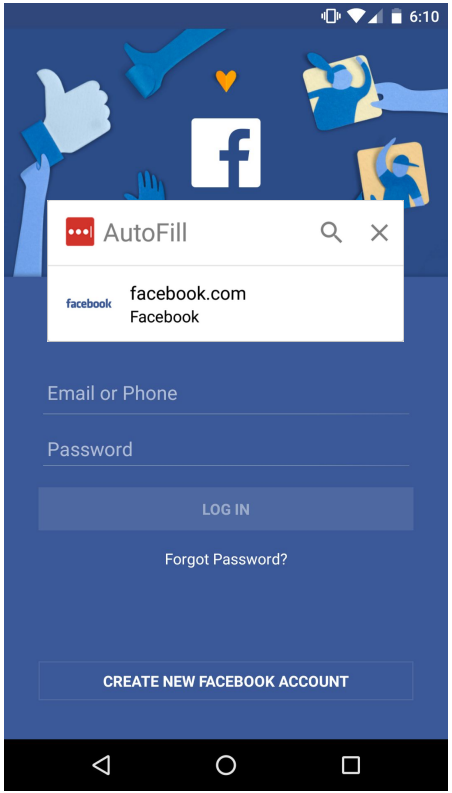
Mobile Phishing 101

- `/proc/*`
- `getRunningTask` API
- Access system log



Phishing Attacks on Modern Android

Mobile Password Managers



How can a password manager know that this app is really linked to facebook.com???

This step is trivial for **browser** password managers, but not on Android...

Three Technologies

- Accessibility Service
- Android Autofill Framework (new in Android 8.0)
- OpenYOLO

In all cases, an app's **package name** is the starting point to map **app** ↔ **website**!

Package Names Can’t Be Trusted

- Nobody is checking / vetting package names
- No trust relation between “package” and “subpackage”
  - E.g., easy to get an app on the official Play Store with “com.facebook.evil” package name
- The only constraints:
  - No two apps can have the same package name on the Play Store
  - No two apps can have the same package name on an Android device at the same time

Real-World Password Managers

Dashlane

- Heuristic to infer the mapping from the package name
  - It splits the package name in components
    - E.g., “aaa.bbb.ccc” →”aaa”, “bbb”, “ccc”
  - For each component, it checks if at least 3 of its characters are contained in the “website” field of each entry

“xxx.face.yyy”→”facebook.com”

“com.inst.lin.ube”→
”instagram.com”, “linkedin.com”, “uber.com”

LastPass

- Heuristic to infer the mapping from the package name
  - It reverses the package name and check for common suffixes with “website” fields of each entry

“com.facebook.evil”→”facebook.com”

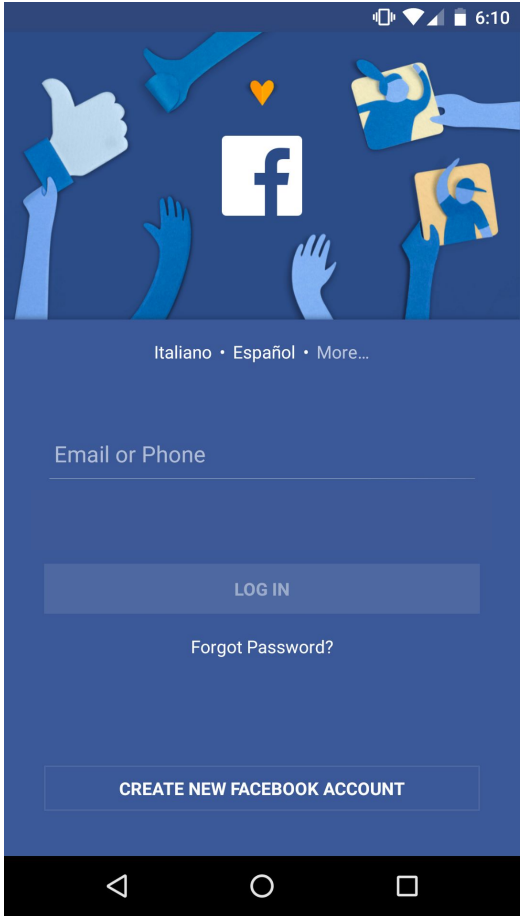
- Crowdsourced mapping
  - Using user-supplied package name ↔ website associations

Keeper

- It takes the
- … it querie

ADDITIONAL INFORMATION

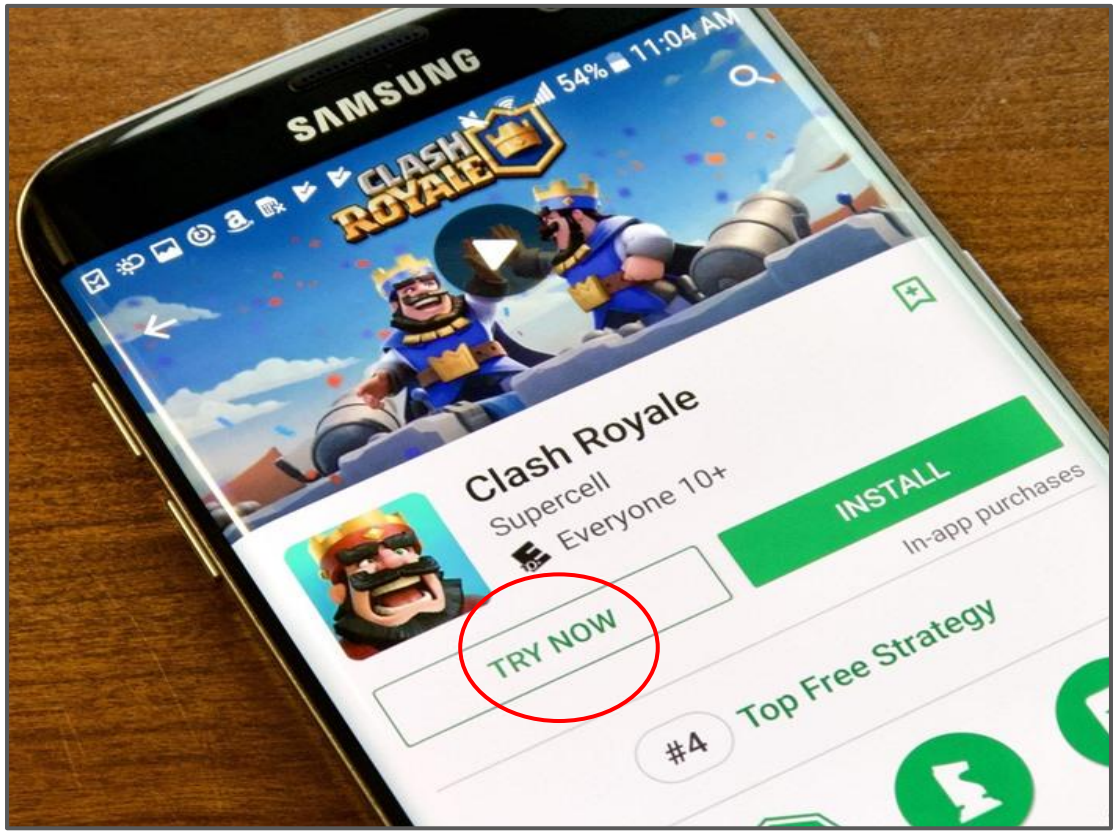
Updated	Size	Installs
September 17, 2018	Varies with device	1,000,000,000+
Current Version	**Requires Android**	**Content Rating**
Varies with device	Varies with device	Teen Learn More
Interactive Elements	**In-app Products**	**Permissions**
Users Interact, Shares Info, Shares Location, Digital Purchases	$0.99 - $399.99 per item	View details
Report	**Offered By**	**Developer**
Flag as inappropriate	Facebook	Visit website android-support@fb.com Privacy Policy 1 Hacker Way Menlo Park, CA 94025


6:10
Italiano • Español • More...
Email or Phone
LOG IN
Forgot Password?
CREATE NEW FACEBOOK ACCOUNT

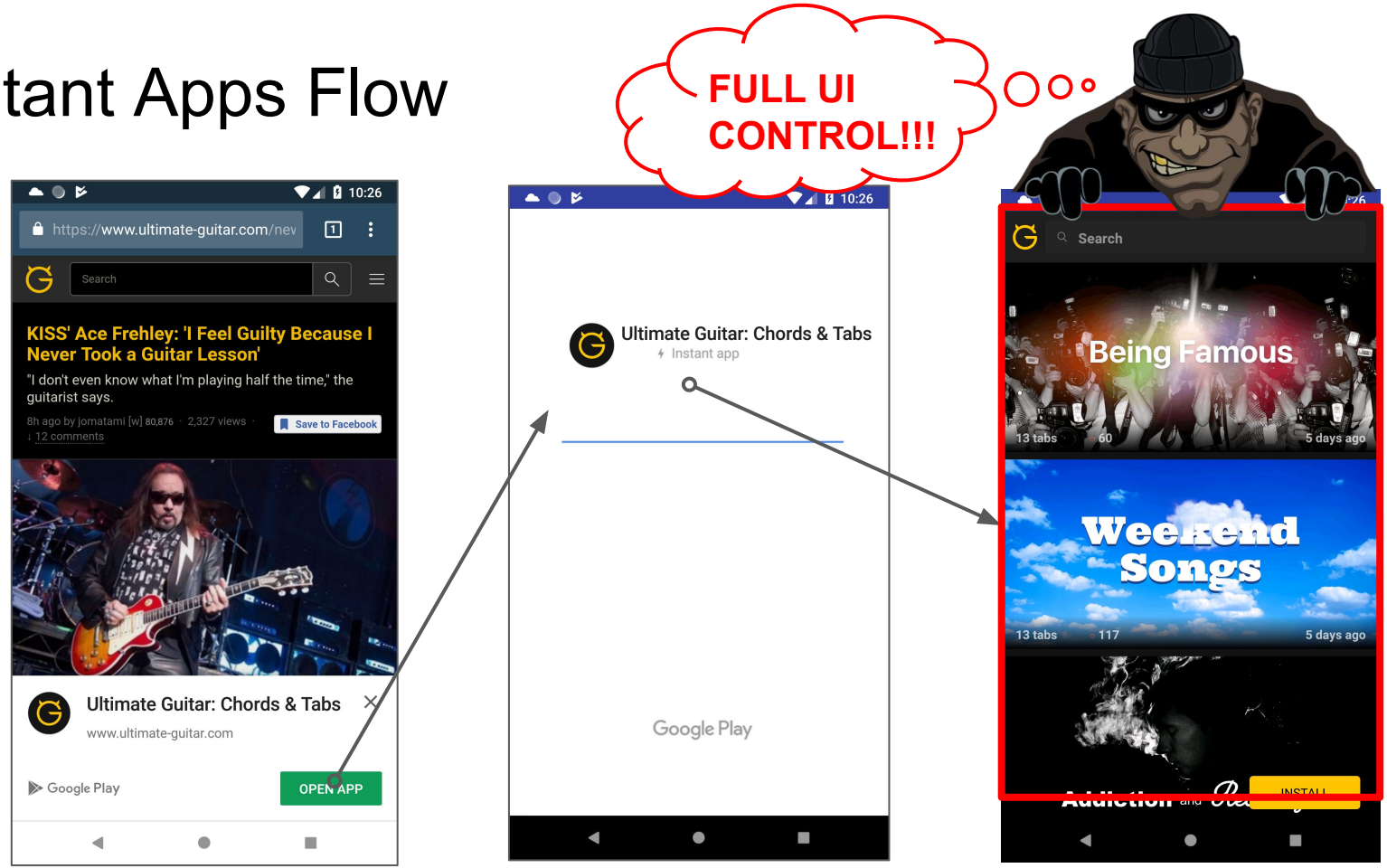
Hidden Fields

- 1x1 pixels
- Foreground color = Background color
- Make fields transparent
- Set “visibility” field to “gone”

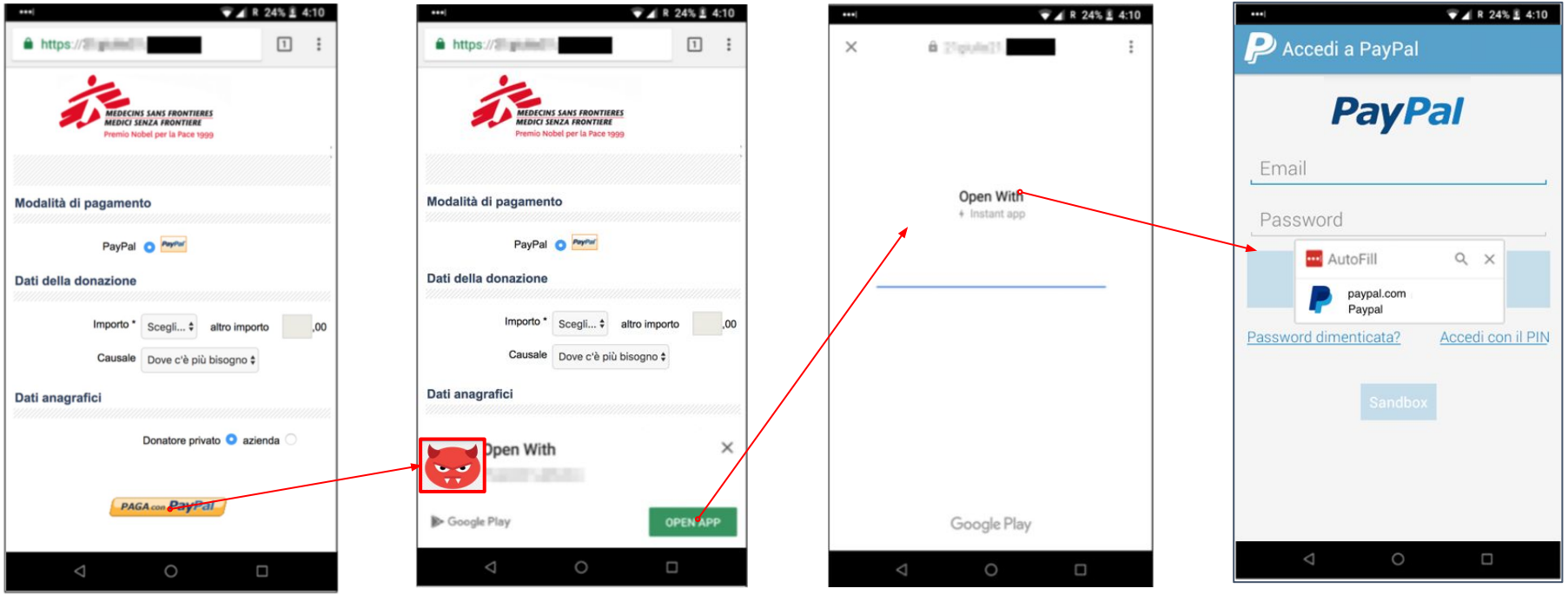
Instant Apps



Instant Apps Flow



End-to-end attack: phishing with few clicks



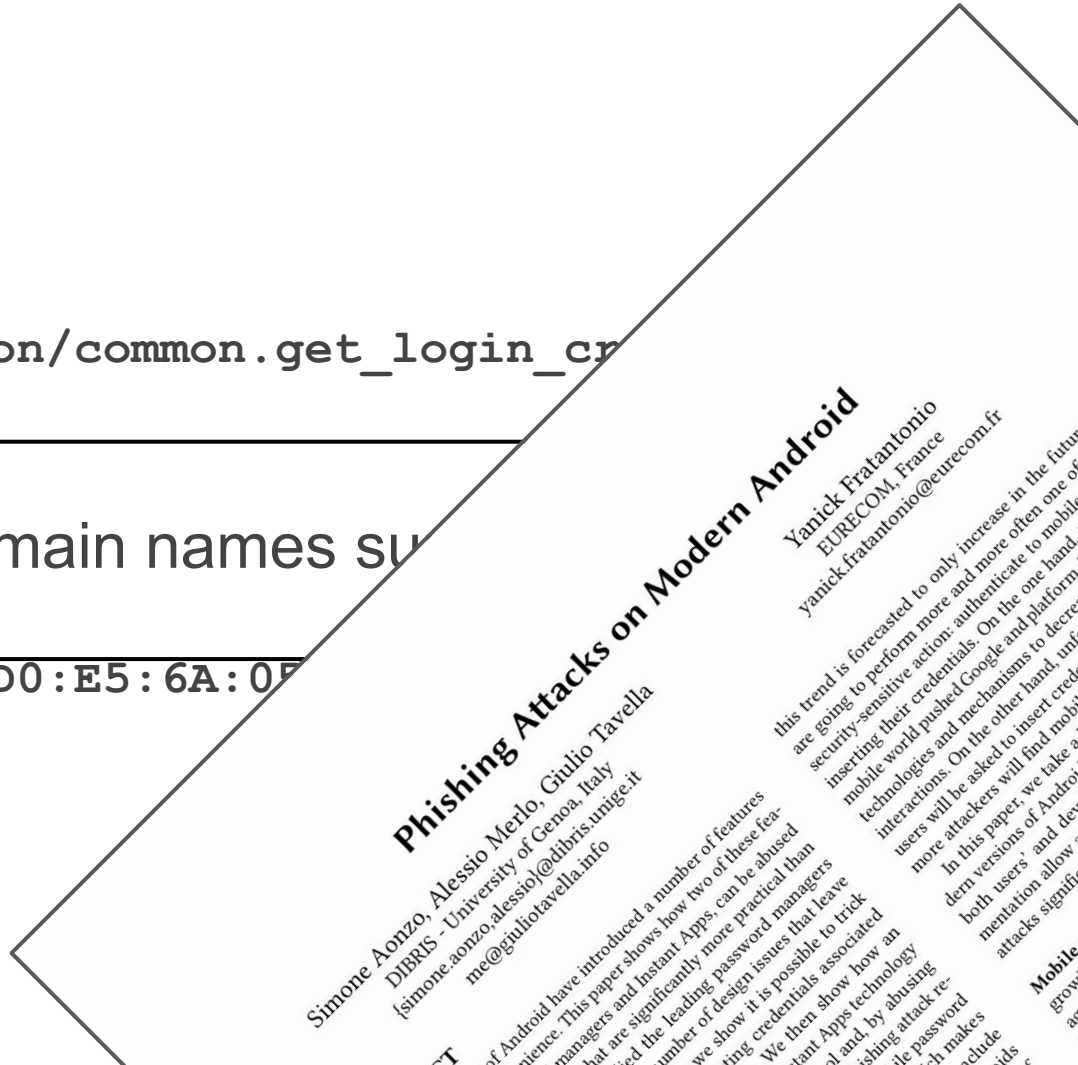
The Right Way™

- Rely on Digital Asset Links (DAL)
- A website can say “apps signed by this certificate are OK”
- https://www.facebook.com/.well-known/assetlinks.json

The Right Way™

```
{
  "relation":
      ["delegate_permission/common.get_login_cr
  "target": {
    "
    "
    "
      "E3:F9:E1:E0:CF:99:D0:E5:6A:0
    ]
  }
}
```

Only ~2% of domain names su



A look at the future

Open problems in mobile UI

- How can I know that I'm interacting with app XYZ?
  - Is it real the facebook app?
- How can the app know that the user intentionally and knowingly clicked on button X?
  - Think about medical devices!
- How can I know that my click has been actually received?
  - If you don't have this guarantee, potential for DOS.
- How can I know that the UI's content is "trusted"?
  - Important for mobile/digital ID (driving licenses, ...)

Android Protected Confirmation

- New API introduced in Android 9.0
  - First very big step towards trusted UI
  - It shows a system-generated popup asking users for confirmation
    - No clickjacking possibilities here
- Security features
  - The UI is actually shown/rendered by Trustzone
    - Even a root attacker can't do much
  - Trustzone is used to generate an attestation code (via cryptography) that encodes "the user has clicked OK + message was XYZ"
  - A network backend can verify the attestation code
    - ⇒ The network backend can be very confident that the user knows about this

UI security is constantly evolving!

- New API in Android: IdentityCredential API
  - Support for "secure" mobile driving licenses (and other docs)
  - Once again based on TrustZone + attestation tokens

- Even the "rules" are changing
  - "Draw on top" permission automatically granted for Play Store apps?
  - [Rumor] In Android Q: apps can't "pop out" from background?
    - I expect (good) impact on adware and simple phishing attacks

Acknowledgments

- My students: Andrea Possemato, Simone Aonzo

- Android security team ⇐ top people

- Security teams of the various password managers (Dashlane, Keeper, LastPass, 1Password)

Related Papers

- "Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop", IEEE S&P'17
- "ClickShield: Are You Hiding Something? Towards Eradicating Clickjacking on Android", CCS'18
- "Phishing Attacks on Modern Android", CCS'18

Thanks!

Yanick Fratantonio

EURECOM

@reyammer

https://reyammer.io

yanick.fratantonio@eurecom.fr

... and stay tuned for CTF-style mobile reversing challs on https://mobisec.reyammer.io/!