When The Web Meets Apps: The Security Pitfalls of In-App Browsing

Cybersecurity Boot Camp 2025

Philipp **Beer**

Security & **Privacy** Research Unit
TU Wien

Who am I?

- **Philipp Beer**
- 1st-year PhD student @ TU Wien
- Mobile Security vs Web Security

✉ philipp.beer@tuwien.ac.at

🦋 @beerphilipp.bsky.social

𝕏 @beerphilipp

Agenda

Background

Threat Models

(WK)WebView

Custom Tabs

TapTrap

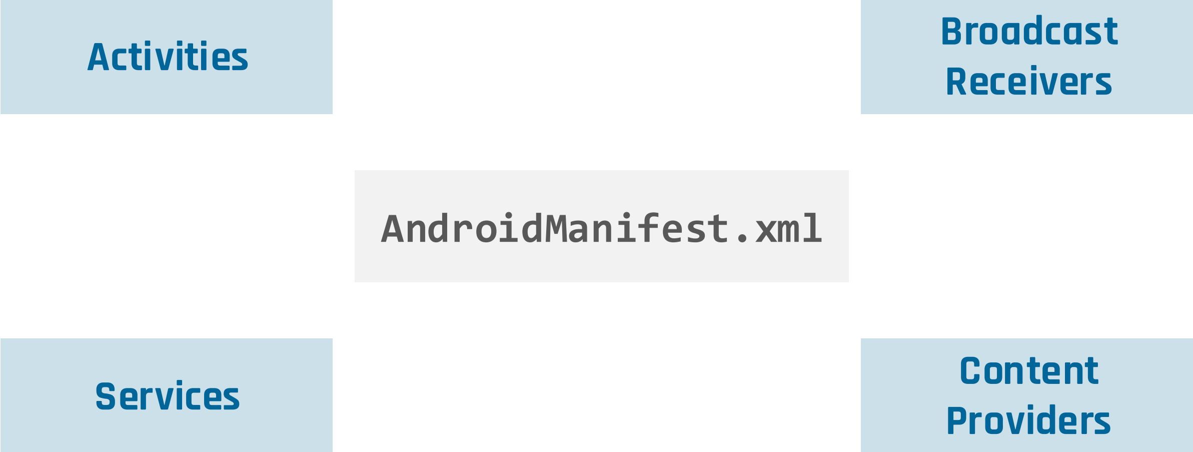
+ DEMO

Go to https://bootcamp25.beerphilipp.com

Background

Android Architecture Basics | Components

An Android app consists of multiple components



Activities

Broadcast Receivers

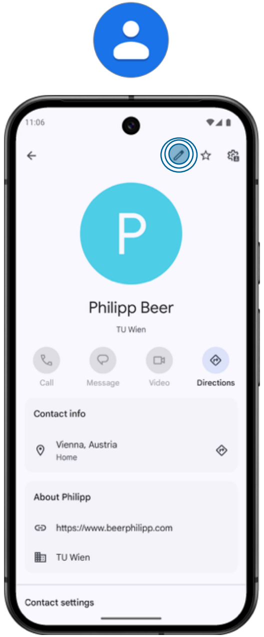
`AndroidManifest.xml`

Services

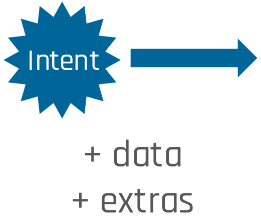
Content Providers

Android Architecture Basics | Activities, Intents, Backstack, Tasks

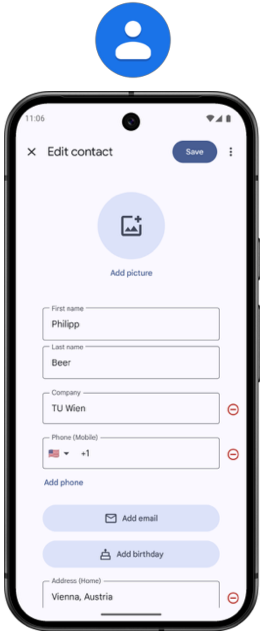
An app consists of multiple activities (a single screen of an app)



PeopleActivity

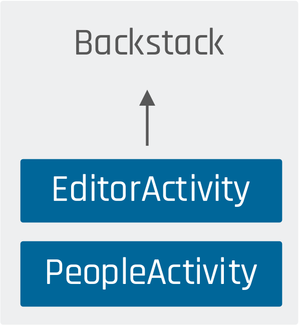


+ data
+ extras



First name: Philipp
Last name: Beer
Company: TU Wien
Phone (Mobile): +1
Address (Home): Vienna, Austria

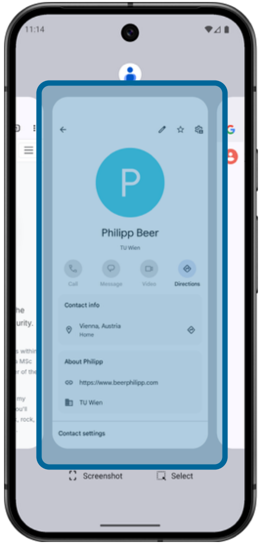
EditorActivity



Backstack

EditorActivity

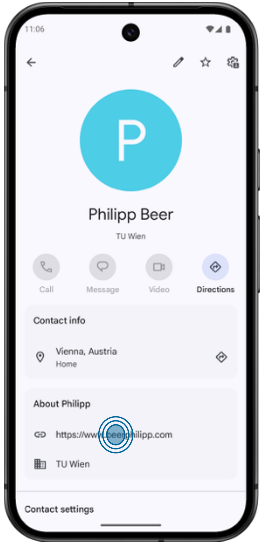
PeopleActivity



Running tasks

Android Architecture Basics | Exported Activities

Intents can also be used to open other apps



Philipp Beer

TU Wien

Contact info

Vienna, Austria
Home

About Philipp

https://www.beerphilipp.com

TU Wien

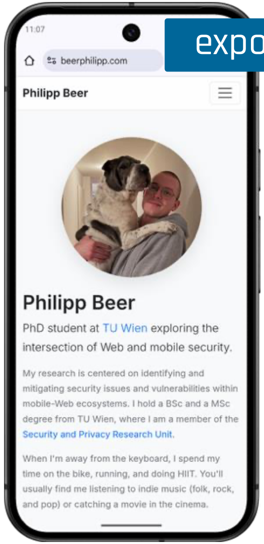
Contact settings

PeopleActivity

Intent

+ data

www.beerphilipp.com



Philipp Beer

PhD student at TU Wien exploring the intersection of Web and mobile security.

My research is centered on identifying and mitigating security issues and vulnerabilities within mobile-Web ecosystems. I hold a BSc and a MSc degree from TU Wien, where I am a member of the Security and Privacy Research Unit.

When I'm away from the keyboard, I spend my time on the bike, running, and doing HIIT. You'll usually find me listening to indie music (folk, rock, and pop) or catching a movie in the cinema.

exported

ChromeTabbedActivity

Exported activities can be launched by other apps

Android Development Basics

Native Android apps are developed in Android Studio

Logic

Java
Kotlin
C/C++

`AndroidManifest.xml`

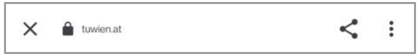
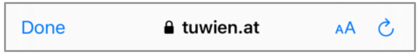
XML

UI

XML

In-App Browsers

Mobile OS's provide different components that app developers can use to display Web content in their apps

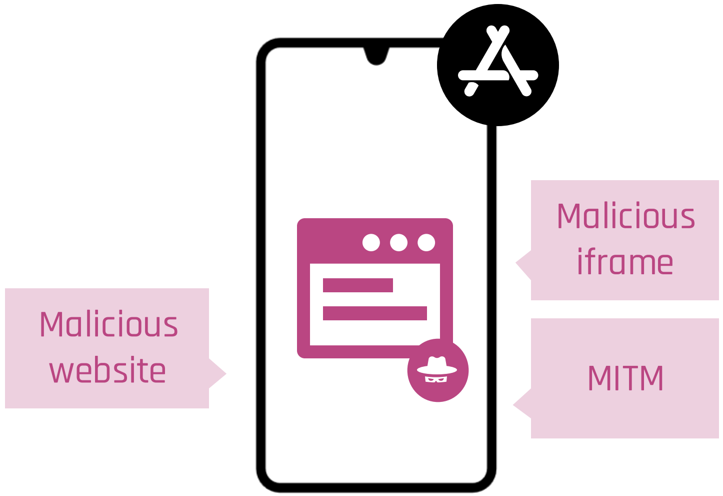
OS	Component	Loads arbitrary websites	Shared state with browser	Web-App Interaction	Browser UI
	WebView	✅	❌	✅	❌
	Custom Tabs	✅	✅	⚠ restricted	
	Trusted Web Activities	❌	✅	⚠ restricted	❌
	3rd-party libraries (e.g. GeckoView)	library-dependent			
iOS	WKWebView	✅	❌	✅	❌
	SFSafariViewController	✅	⚠ restricted	⚠ restricted	

Threat Models

Threat Models

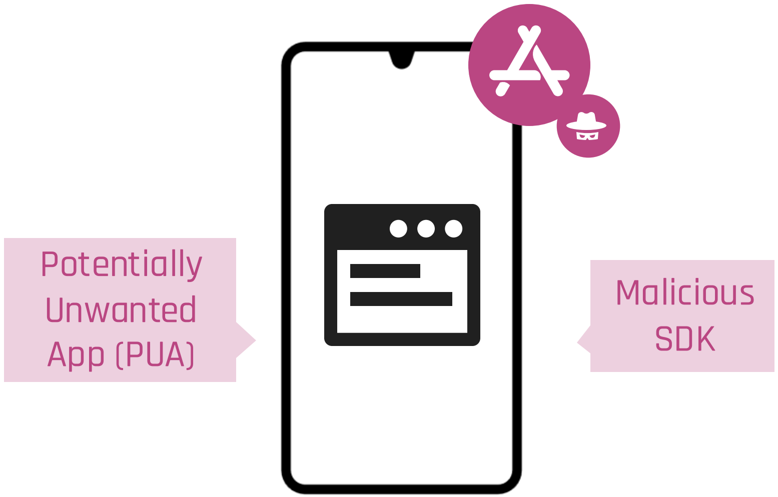
Web-based Attacker

Malicious web content is loaded inside a benign app



App-based Attacker

A benign website is loaded in a malicious app

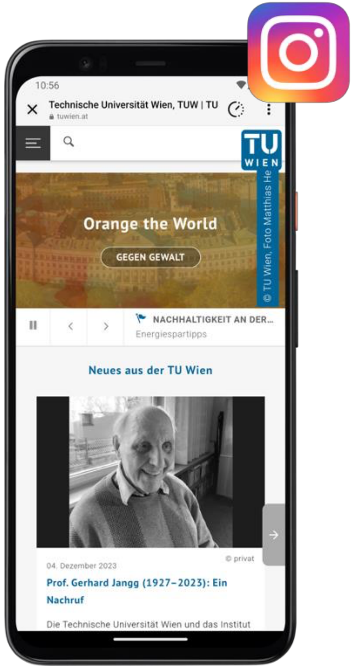


WebView & WKWebView

(WK)WebView

- system components
  - Android WebView: Chromium
  - iOS WKWebView: WebKit
- **no shared browsing data** (e.g., cookies, cache, etc.) with the underlying browsers or other apps
- support a **high level of web-app interaction**
- used by apps to
  - show websites
  - display ads
  - build hybrid apps (e.g., Cordova)
  - build browsers
  - ...

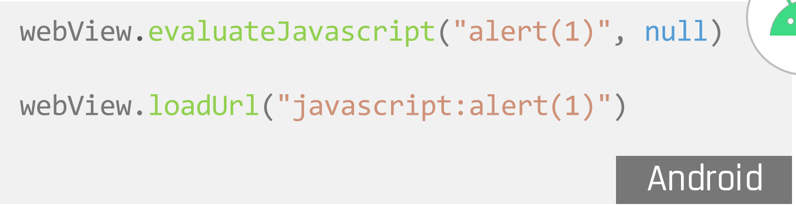
About 56% of apps use Android WebView


10:56
Technische Universität Wien, TUW | TU
tuwien.at
TU WIEN
© TU Wien, Foto Matthias He
Orange the World
GEGEN GEWALT
NACHHALTIGKEIT AN DER...
Energiespartipps
Neues aus der TU Wien
© privat
04. Dezember 2023
Prof. Gerhard Jangg (1927–2023): Ein Nachruf
Die Technische Universität Wien und das Institut

In-app browser on Instagram powered by WebView

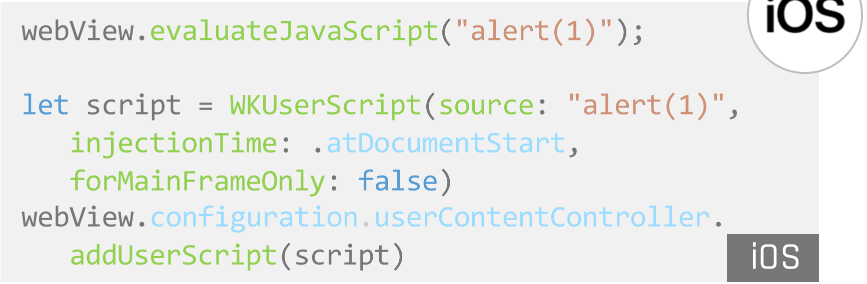
App-to-Web Interaction | JS Injection

An app can inject JavaScript code into a website



```
webView.evaluateJavascript("alert(1)", null)

webView.loadUrl("javascript:alert(1)")
```

Android



```
webView.evaluateJavaScript("alert(1)");

let script = WKUserScript(source: "alert(1)",
   injectionTime: .atDocumentStart,
   forMainFrameOnly: false)
webView.configuration.userContentController.
   addUserScript(script)
```

iOS

Problem: A PUA can inject JavaScript code to modify the website, monitor user interactions, and steal user data

TikTok on iOS injected JS code into every website and subscribed to the `keypress` and `keydown` events that could be used to record all user input (2022)

See this blog post for more

App-to-Web Interaction | Access & Modification of Cookies

An app can read and write cookies of a website

```
val cookieManager = CookieManager.getInstance()

cookieManager.setCookie(
   "https://example.com", "a=b")

val cookies = cookieManager.getCookie(
   "https://example.com")
```

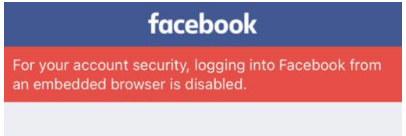
Android

```
let store = webView.configuration.
websiteDataStore.httpCookieStore

store.setCookie(HTTPCookie(properties:
  [.domain: "example.com", .path: "/", .name: "a",
   .value: "1", .secure: "TRUE"])!)

store.getAllCookies { }
```

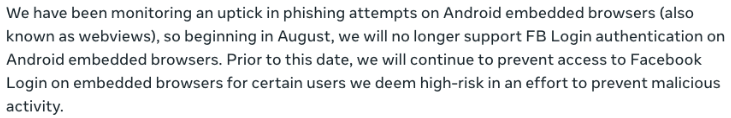
iOS

Problem: A PUA can steal cookies, hijack the user‘s session, or perform session swapping attacks

Google and Facebook block logins from (WK)WebView



For your account security, logging into Facebook from an embedded browser is disabled.



We have been monitoring an uptick in phishing attempts on Android embedded browsers (also known as webviews), so beginning in August, we will no longer support FB Login authentication on Android embedded browsers. Prior to this date, we will continue to prevent access to Facebook Login on embedded browsers for certain users we deem high-risk in an effort to prevent malicious activity.

App-to-Web Interaction | Access & Modification of Cookies

```
val cookieMana

cookieManager.
   "https://ex

val cookies =
   "https://ex
```

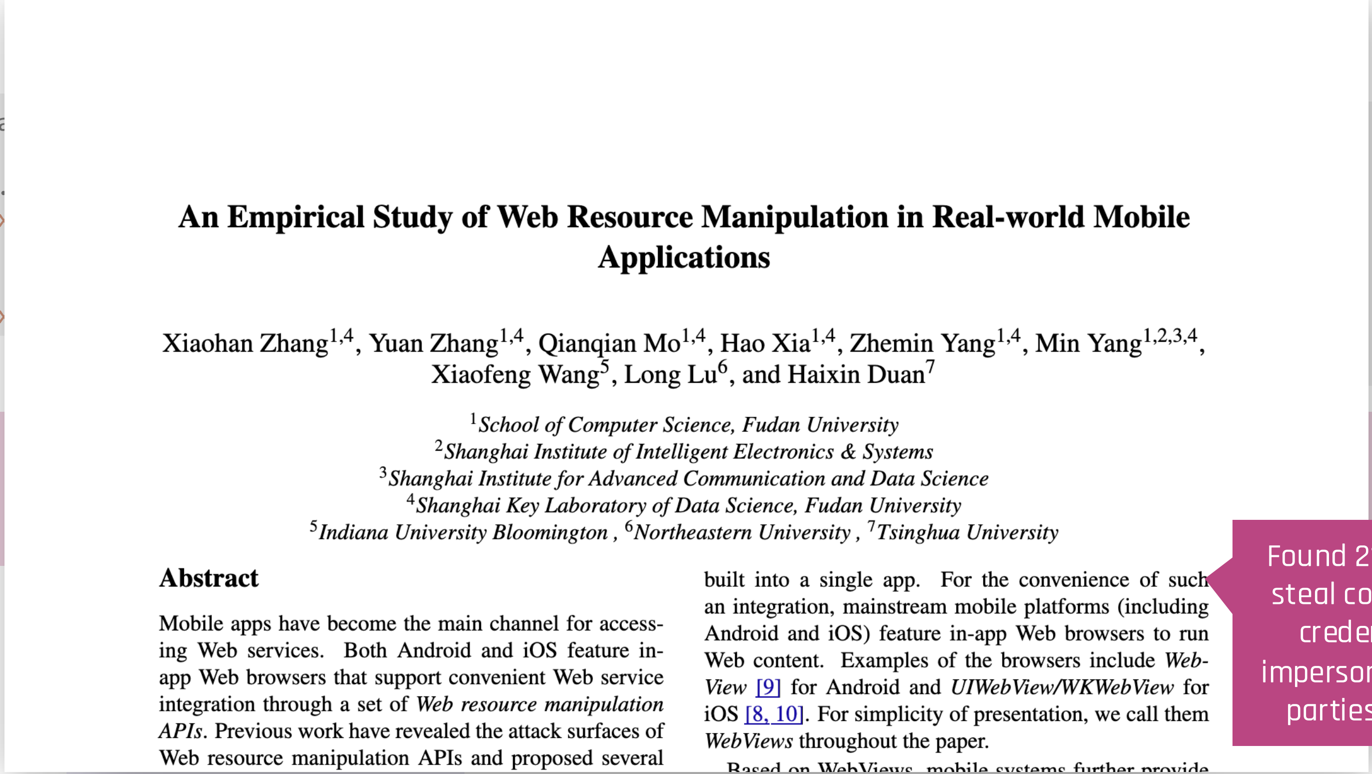
iOS

```
"a",
```

iOS

Problem: A
attacks

Google and



An Empirical Study of Web Resource Manipulation in Real-world Mobile Applications

Xiaohan Zhang[1,4], Yuan Zhang[1,4], Qianqian Mo[1,4], Hao Xia[1,4], Zhemin Yang[1,4], Min Yang[1,2,3,4], Xiaofeng Wang[5], Long Lu[6], and Haixin Duan[7]

[1] *School of Computer Science, Fudan University*
[2] *Shanghai Institute of Intelligent Electronics & Systems*
[3] *Shanghai Institute for Advanced Communication and Data Science*
[4] *Shanghai Key Laboratory of Data Science, Fudan University*
[5] *Indiana University Bloomington* , [6] *Northeastern University* , [7] *Tsinghua University*

Abstract

Mobile apps have become the main channel for accessing Web services. Both Android and iOS feature in-app Web browsers that support convenient Web service integration through a set of *Web resource manipulation APIs*. Previous work have revealed the attack surfaces of Web resource manipulation APIs and proposed several

built into a single app. For the convenience of such an integration, mainstream mobile platforms (including Android and iOS) feature in-app Web browsers to run Web content. Examples of the browsers include *WebView* [9] for Android and *UIWebView/WKWebView* for iOS [8, 10]. For simplicity of presentation, we call them *WebViews* throughout the paper.

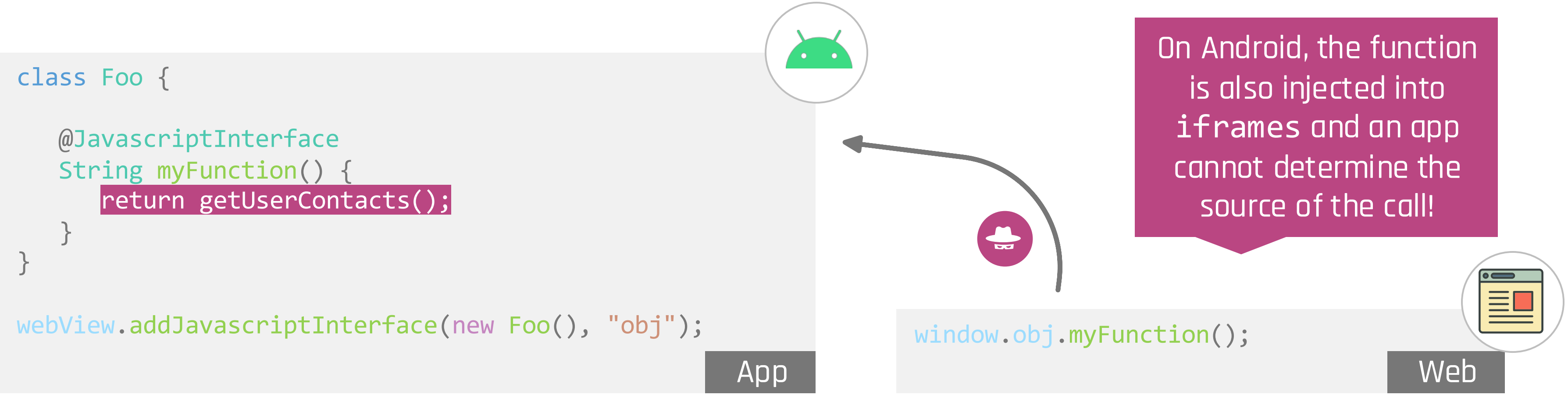
Based on WebViews, mobile systems further provide

activity.

Found 21 apps that steal cookies, user credentials, or impersonate relying parties in OAuth

Web-to-App Interaction | Calling Native Functions ("JS Bridge")

Websites loaded in a (WK)WebView can call native functions defined in the app



```
class Foo {

   @JavascriptInterface
   String myFunction() {
      return getUserContacts();
   }
}

webView.addJavascriptInterface(new Foo(), "obj");
```

```
window.obj.myFunction();
```

On Android, the function is also injected into `iframes` and an app cannot determine the source of the call!

Problem: A malicious website can leak sensitive information or perform unwanted actions in the context of the app

Permission Enforcement

WebView has no built-in browser permission prompt

WebView delegates the decision to the app

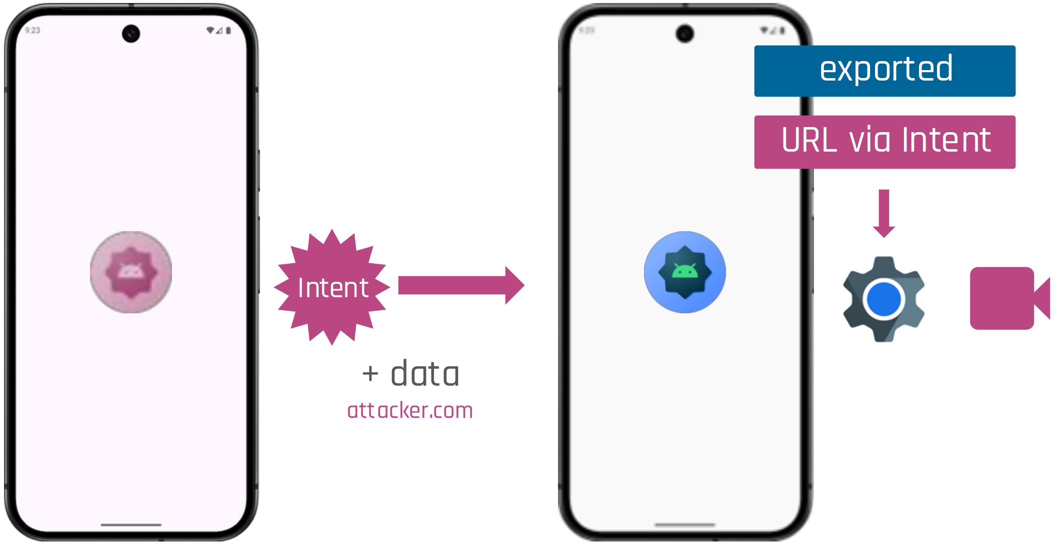
Problem: Apps may be too lax and allow access to all sites when loaded

```
webView.webChromeClient = object : WebChromeClient() {

   override fun onPermissionRequest(
      request: PermissionRequest) {
      // Grant or deny camera/microphone permission
      request.deny() // or
      request.grant(request.resources)
   }

   override fun onGeolocationPermissionShowPrompt(
      origin: String,
      callback: Callback) {
      callback.invoke(
         origin,
         /*allow*/ true,
         /*retain*/ false)
   }
}
```

App

Permission Enforcement | PUA

PUAs can abuse a vulnerable app

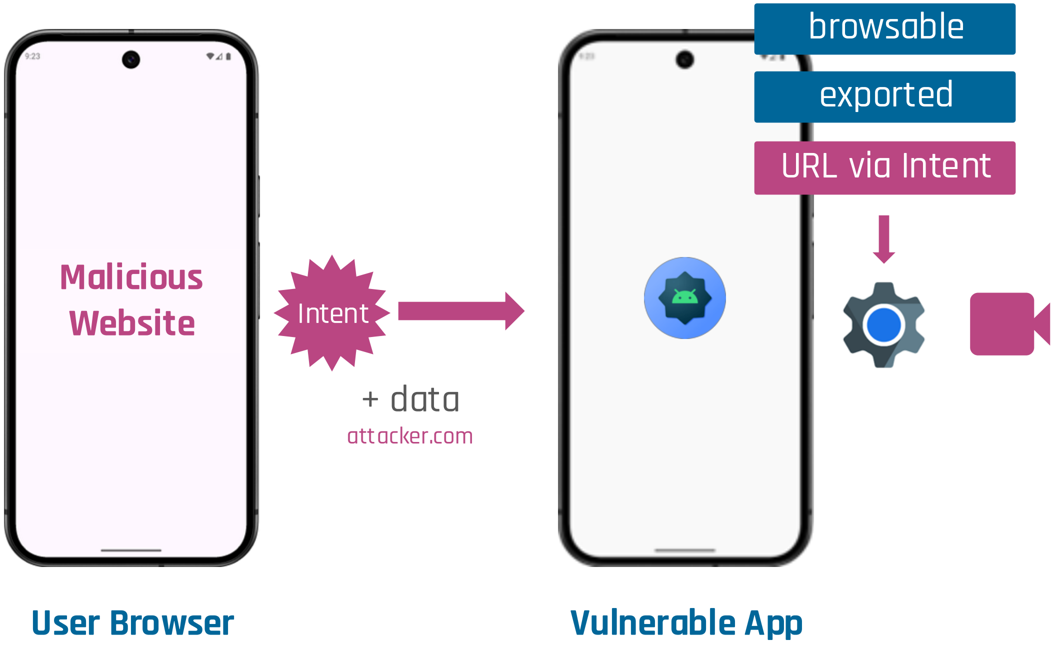


PUA

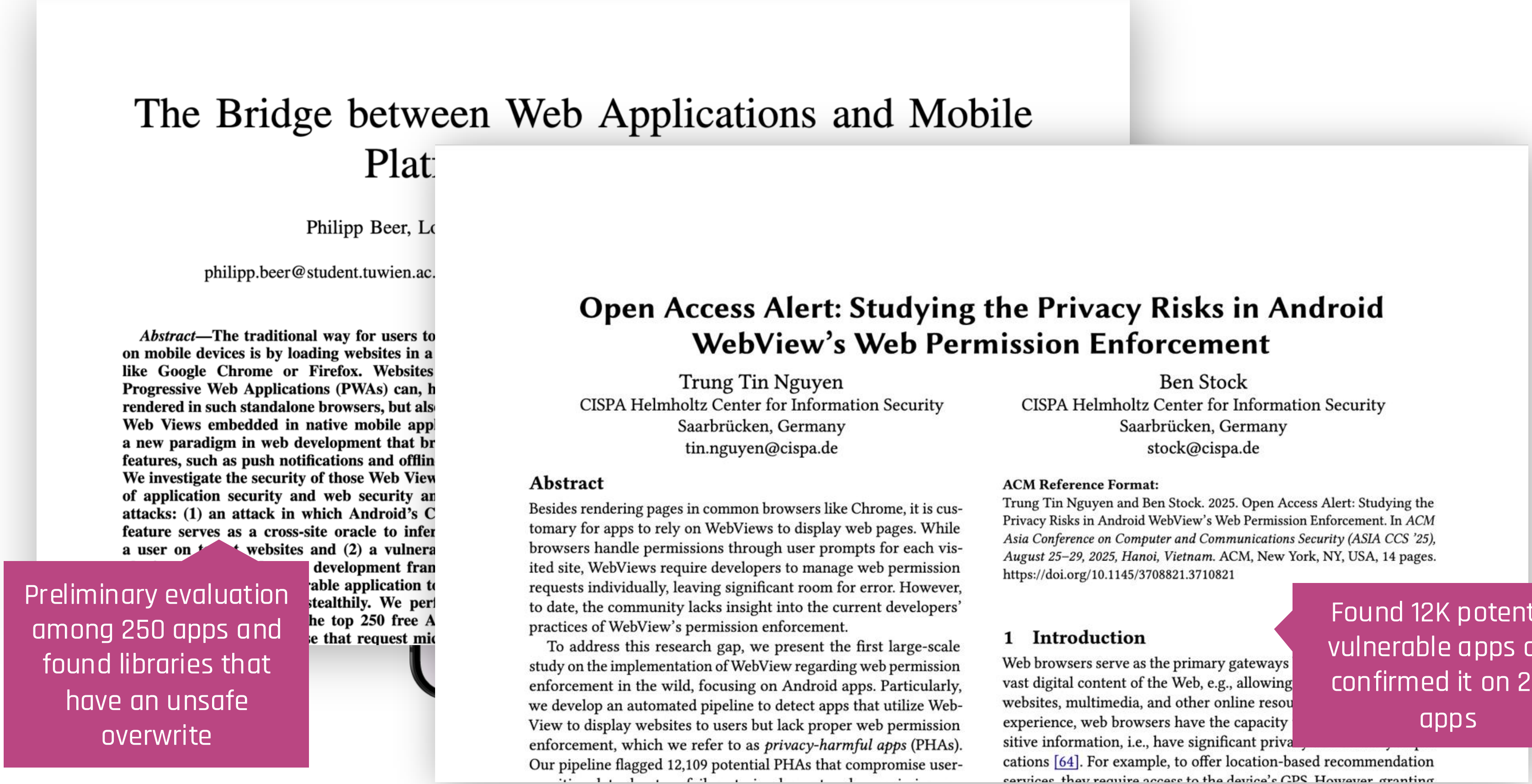
Vulnerable App

Permission Enforcement | Malicious Website

PUAs can abuse a vulnerable app



Permission Enforcement | Malicious Website



The Bridge between Web Applications and Mobile
Plat

Philipp Beer, Lo

philipp.beer@student.tuwien.ac.

Abstract—The traditional way for users to
on mobile devices is by loading websites in a
like Google Chrome or Firefox. Websites
Progressive Web Applications (PWAs) can, h
rendered in such standalone browsers, but als
Web Views embedded in native mobile appl
a new paradigm in web development that br
features, such as push notifications and offlin
We investigate the security of those Web View
of application security and web security an
attacks: (1) an attack in which Android's C
feature serves as a cross-site oracle to infer
a user on t websites and (2) a vulnera
development fran
able application to
stealthily. We perf
he top 250 free A
e that request mic

Open Access Alert: Studying the Privacy Risks in Android WebView's Web Permission Enforcement

Trung Tin Nguyen
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
tin.nguyen@cispa.de

Ben Stock
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
stock@cispa.de

Abstract

Besides rendering pages in common browsers like Chrome, it is customary for apps to rely on WebViews to display web pages. While browsers handle permissions through user prompts for each visited site, WebViews require developers to manage web permission requests individually, leaving significant room for error. However, to date, the community lacks insight into the current developers' practices of WebView's permission enforcement.

To address this research gap, we present the first large-scale study on the implementation of WebView regarding web permission enforcement in the wild, focusing on Android apps. Particularly, we develop an automated pipeline to detect apps that utilize WebView to display websites to users but lack proper web permission enforcement, which we refer to as *privacy-harmful apps* (PHAs). Our pipeline flagged 12,109 potential PHAs that compromise user-

ACM Reference Format:
Trung Tin Nguyen and Ben Stock. 2025. Open Access Alert: Studying the Privacy Risks in Android WebView's Web Permission Enforcement. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '25), August 25–29, 2025, Hanoi, Vietnam.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3708821.3710821

1 Introduction

Web browsers serve as the primary gateways
vast digital content of the Web, e.g., allowing
websites, multimedia, and other online resou
experience, web browsers have the capacity
sitive information, i.e., have significant priva
cations [64]. For example, to offer location-based recommendation

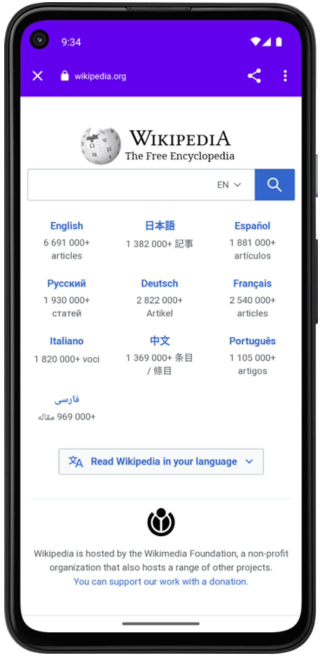
Preliminary evaluation among 250 apps and found libraries that have an unsafe overwrite

Found 12K potential vulnerable apps and confirmed it on 2.2k apps

Custom Tabs

Custom Tabs

- A Custom Tab is provided by the underlying browser
  - e.g., Chrome, Firefox, Brave
- Unlike `(WK)WebView`, it **shares state** with the browser
  - e.g., cookies, service workers, cache etc. are shared
- Often used for Authentication/Authorization purposes
  - OAuth 2.0 standard recommends using Custom Tabs on Android
- Web-App interaction is **highly limited**



9:34

wikipedia.org

Wikipedia
The Free Encyclopedia

EN

English
6 691 000+ articles

日本語
1 382 000+ 記事

Español
1 881 000+ artículos

Русский
1 930 000+ статей

Deutsch
2 822 000+ Artikel

Français
2 540 000+ articles

Italiano
1 820 000+ voci

中文
1 369 000+ 条目 / 條目

Português
1 105 000+ artigos

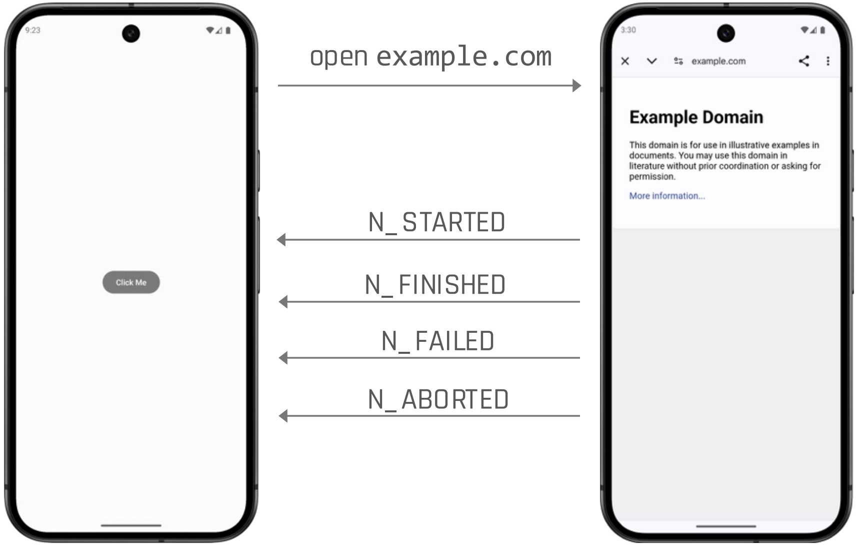
فارسی
969 000+ مقاله

Read Wikipedia in your language

Wikipedia is hosted by the Wikimedia Foundation, a non-profit organization that also hosts a range of other projects.
You can support our work with a donation.

Custom Tabs | Callbacks

Callbacks keep the app informed about the status of website loading



Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

App

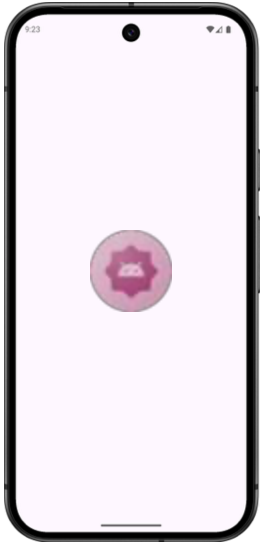
Custom Tab

Callbacks depend on behavior of the website

Behavior of the website depends on user's state

Cross-Context State Inference | Example

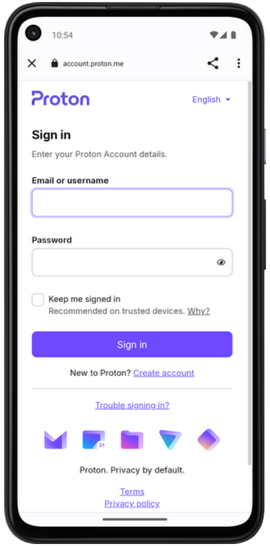
Callbacks can be abused as a cross-context oracle



Open Proton →

← N_STARTED

← N_FINISHED



Sign in

Enter your Proton Account details.

Email or username

Password

□ Keep me signed in
Recommended on trusted devices. Why?

Sign in

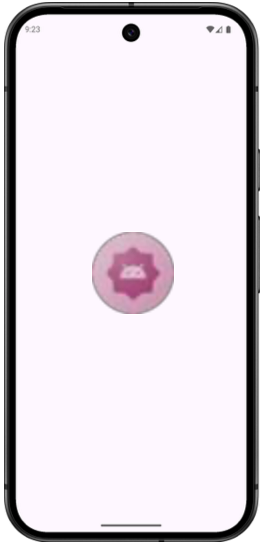
New to Proton? Create account

Trouble signing in?

Proton. Privacy by default.

Terms
Privacy policy

Not logged in



Open Proton →

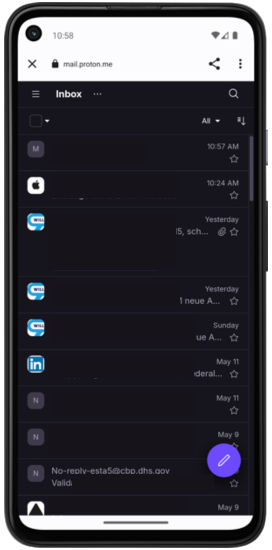
← N_STARTED

← N_FINISHED

redir

← **N_STARTED**

← **N_FINISHED**



M [redacted] 10:57 AM

[redacted] 10:24 AM

[redacted] Yesterday 15, sch...

[redacted] Yesterday 1 neue A...

[redacted] Sunday ue A...

[redacted] May 11 deral...

N [redacted] May 11

N [redacted] May 9

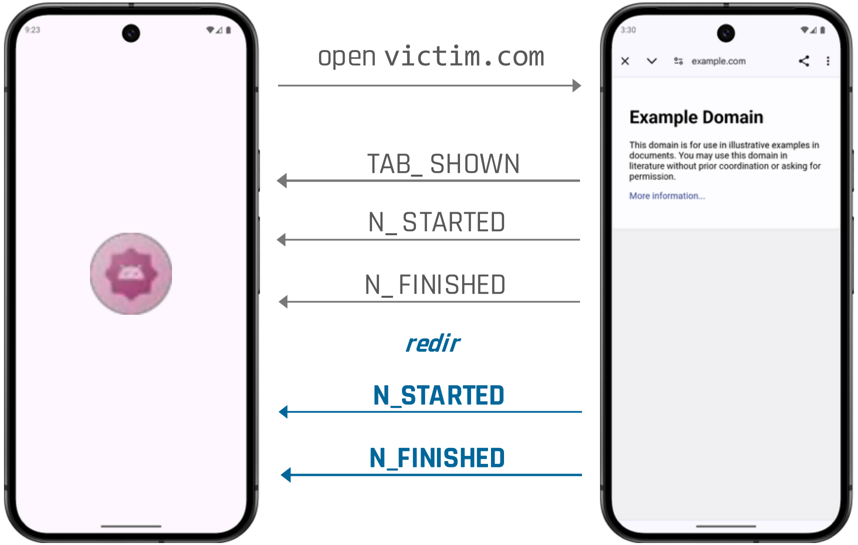
N No-reply-esta5@cbp.dhs.gov
Valid

May 9

Logged in

Cross-Context State Inference | Detectable Behavior

Callbacks can be abused as a cross-context oracle



open `victim.com`

TAB_ SHOWN

N_ STARTED

N_ FINISHED

redir

N_STARTED

N_FINISHED

Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

PUA

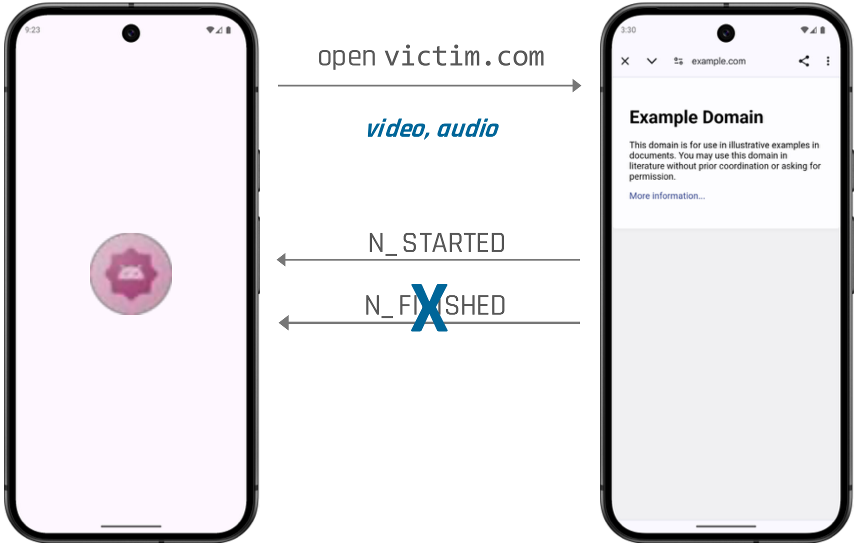
Custom Tab

Redirection

JS and meta redirections trigger STARTED and FINISHED events

Cross-Context State Inference | Detectable Behavior

Callbacks can be abused as a cross-context oracle



Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

PUA

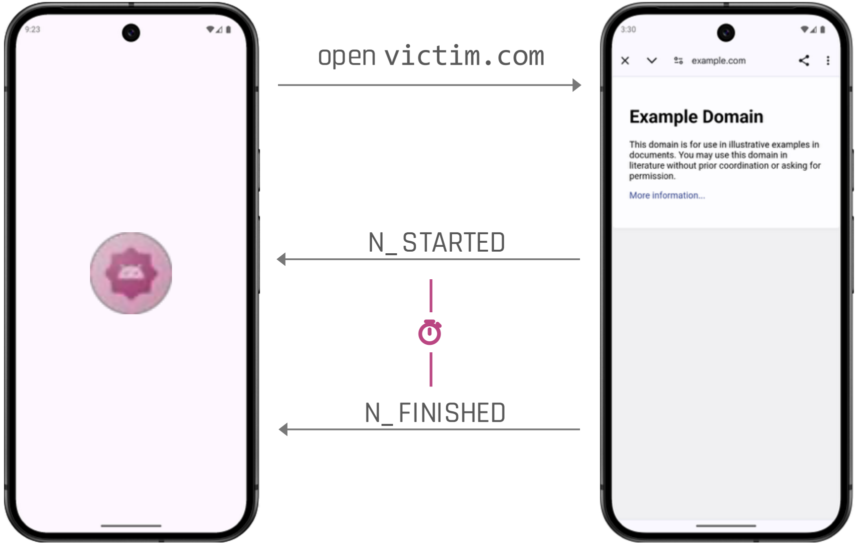
Custom Tab

Content Type

videos and audios do not trigger the FINISHED event

Cross-Context State Inference | Detectable Behavior

Callbacks can be abused as a cross-context oracle



open victim.com

N_STARTED

N_FINISHED

example.com

Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

PUA

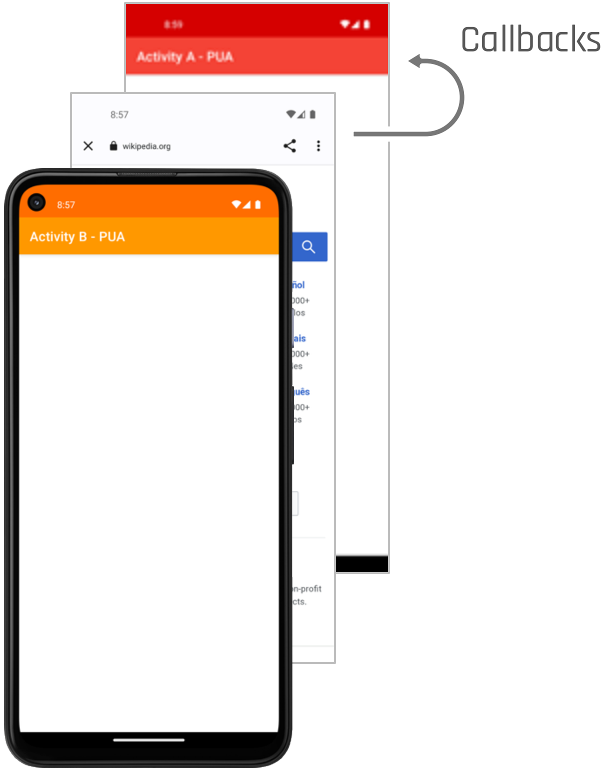
Custom Tab

Timing

Measure the time between the STARTED and FINISHED events

Cross-Context State Inference | Hiding

A Custom Tab can be hidden by overlaying it with another activity



Cross-Context State Inference | Unique Characteristics

Framing Protections

X-Frame-Options, CSP frame-ancestors

SameSite Strict Cookies

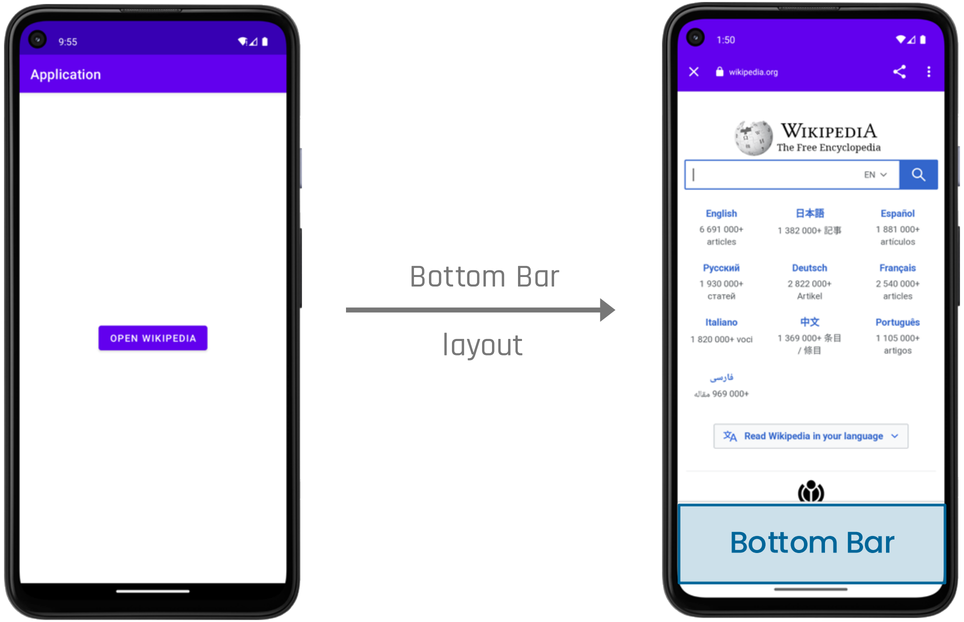
SameSite strict cookies sent on navigation

<109 <1.48 <110

SameSite Lax cookies are still sent!

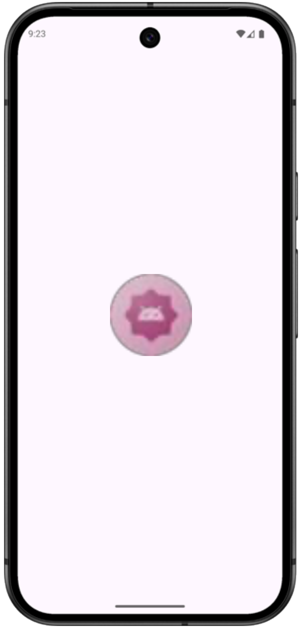
Bottom Bar | Custom Layout

An app can fully customize the container at the bottom of the Custom Tab

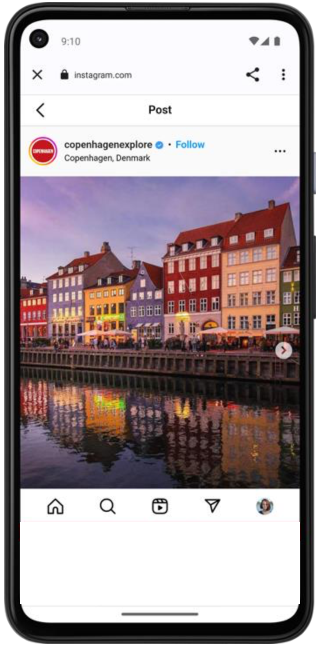


Bottom Bar Problems | Phishing

The bottom bar can be abused for phishing



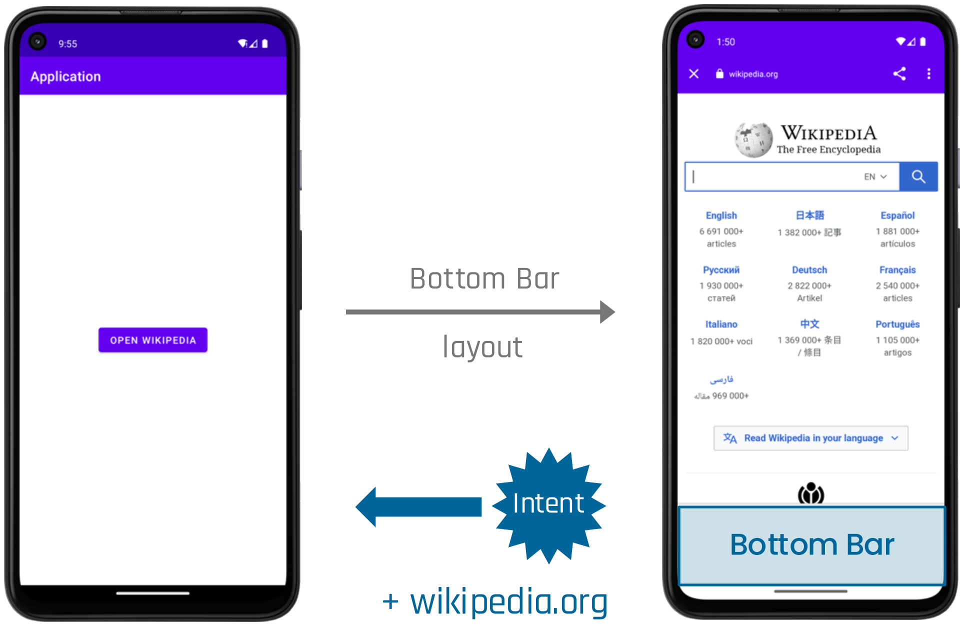
PUA



Custom Tab

Bottom Bar | Intent

An app can fully customize the container at the bottom of the Custom Tab



Bottom Bar

layout

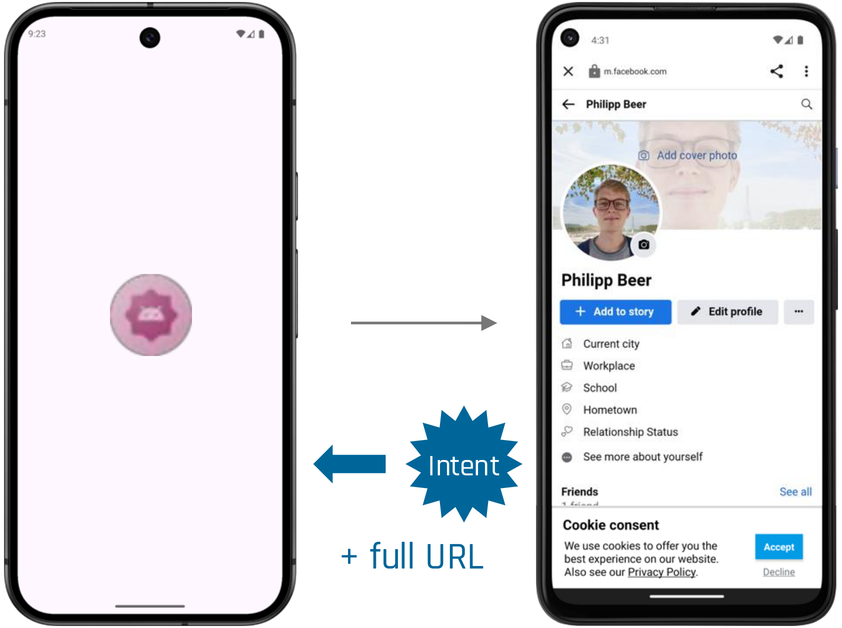
Intent

+ wikipedia.org

Bottom Bar

Bottom Bar Problems | Information Leak

The bottom bar can be abused to leak user information



9:23

4:31

m.facebook.com

Philipp Beer

Add cover photo

Philipp Beer

+ Add to story | Edit profile | ···

- Current city
- Workplace
- School
- Hometown
- Relationship Status
- See more about yourself

Friends — See all

Cookie consent

We use cookies to offer you the best experience on our website. Also see our Privacy Policy.

Accept

Decline

Intent

+ full URL

PUA

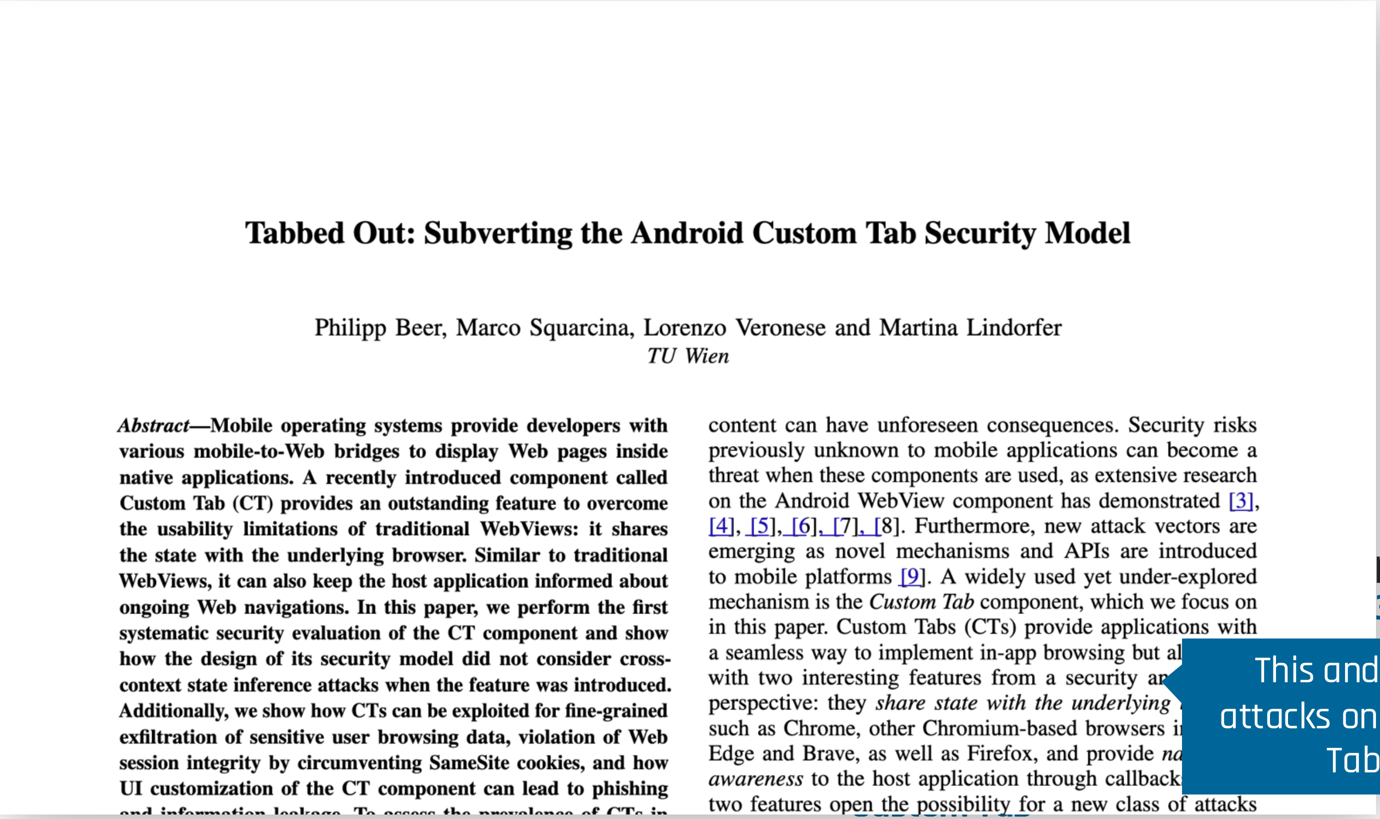
Custom Tab

```
m.facebook.com
/profile
```

↓

```
m.facebook.com/
profile.php/?id=
100080788234879
```

Bottom Bar Problems | Information Leak



Tabbed Out: Subverting the Android Custom Tab Security Model

Philipp Beer, Marco Squarcina, Lorenzo Veronese and Martina Lindorfer
TU Wien

Abstract—**Mobile operating systems provide developers with various mobile-to-Web bridges to display Web pages inside native applications. A recently introduced component called Custom Tab (CT) provides an outstanding feature to overcome the usability limitations of traditional WebViews: it shares the state with the underlying browser. Similar to traditional WebViews, it can also keep the host application informed about ongoing Web navigations. In this paper, we perform the first systematic security evaluation of the CT component and show how the design of its security model did not consider cross-context state inference attacks when the feature was introduced. Additionally, we show how CTs can be exploited for fine-grained exfiltration of sensitive user browsing data, violation of Web session integrity by circumventing SameSite cookies, and how UI customization of the CT component can lead to phishing and information leakage. To assess the prevalence of CTs in**

content can have unforeseen consequences. Security risks previously unknown to mobile applications can become a threat when these components are used, as extensive research on the Android WebView component has demonstrated [3], [4], [5], [6], [7], [8]. Furthermore, new attack vectors are emerging as novel mechanisms and APIs are introduced to mobile platforms [9]. A widely used yet under-explored mechanism is the *Custom Tab* component, which we focus on in this paper. Custom Tabs (CTs) provide applications with a seamless way to implement in-app browsing but al with two interesting features from a security ar perspective: they *share state with the underlying* such as Chrome, other Chromium-based browsers i Edge and Brave, as well as Firefox, and provide *na awareness* to the host application through callback two features open the possibility for a new class of attacks

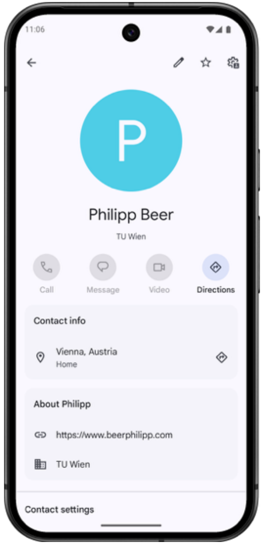
.com

.com/
p/?id=
34879

This and more attacks on Custom Tabs

TapTrap

Activity Transitions



11:06

Philipp Beer

TU Wien

Call Message Video Directions

Contact info

Vienna, Austria
Home

About Philipp

https://www.beerphilipp.com

TU Wien

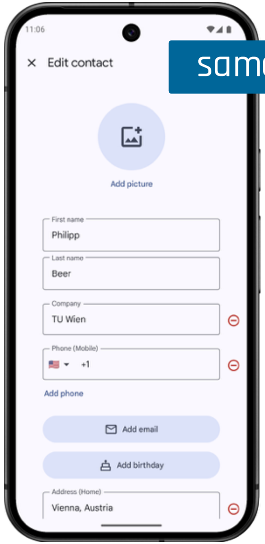
Contact settings

PeopleActivity

Intent

+

Animation



11:06

Edit contact

same-task

Add picture

First name
Philipp

Last name
Beer

Company
TU Wien

Phone (Mobile)
+1

Add phone

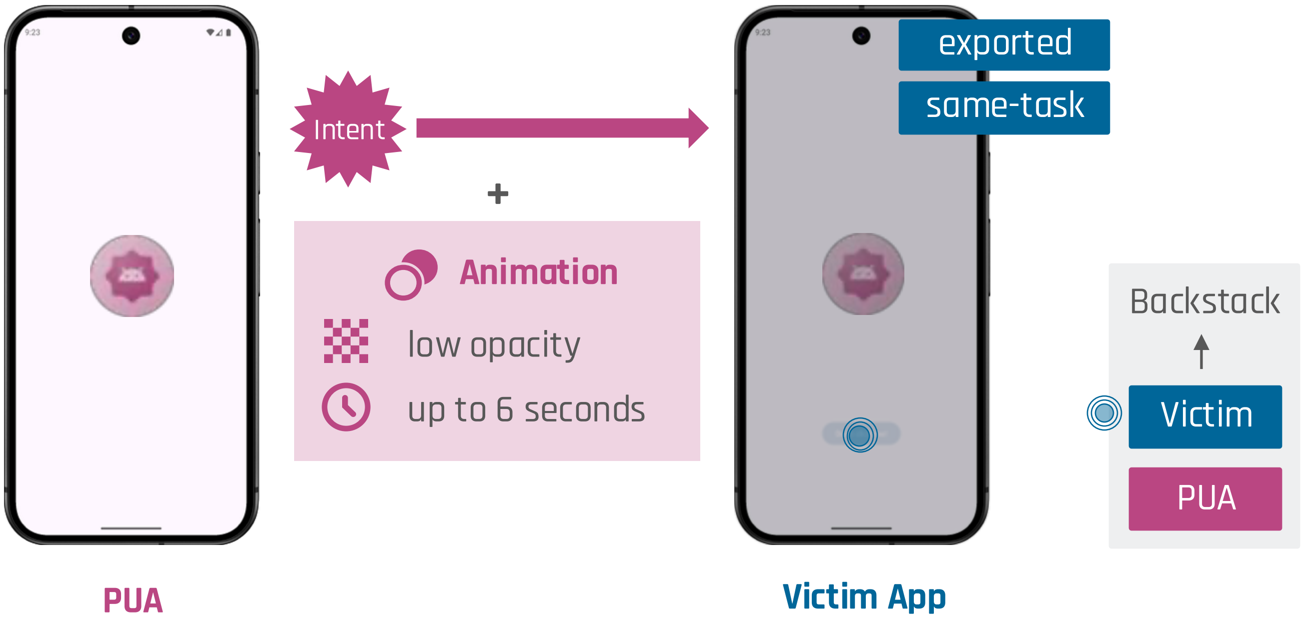
Add email

Add birthday

Address (Home)
Vienna, Austria

EditorActivity

TapTrap | Mechanism



TapTrap | Implications

Browser

Fixed 3 months ago!

Permission Bypass

Load **attacker-controlled website** in a Custom Tab that requests sensitive permission

Web Clickjacking

Open **victim website** in a Custom Tab and lure users into clicking sensitive button, e.g., “pay now”

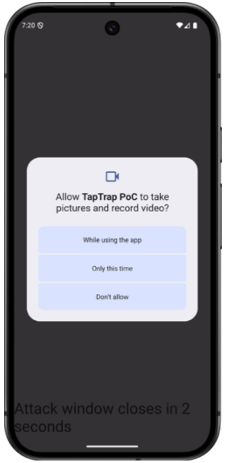
3rd Party Apps

Analysis of ~100K apps from the Play Store

76% of apps are vulnerable (contain a vulnerable activity)
7% of all activities are vulnerable

System Apps and Dialogs

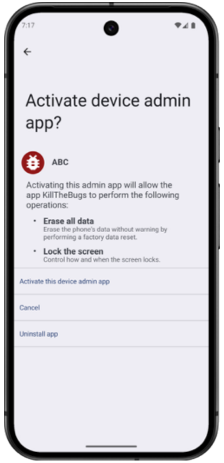
Still vulnerable



Allow **TapTrap PoC** to take pictures and record video?

Attack window closes in 2 seconds

Bypass runtime permissions



Activate device admin app?

Activating this admin app will allow the app KillTheBugs to perform the following operations:

- **Erase all data**
  Erase the phone's data without warning by performing a factory data reset.
- **Lock the screen**
  Control how and when the screen locks.

Device erasure

TapTrap | Implications

Browser

Permission E

Load **attacker-**
requests sensi

Web Clickjac

Open **victim w**
clicking sensiti

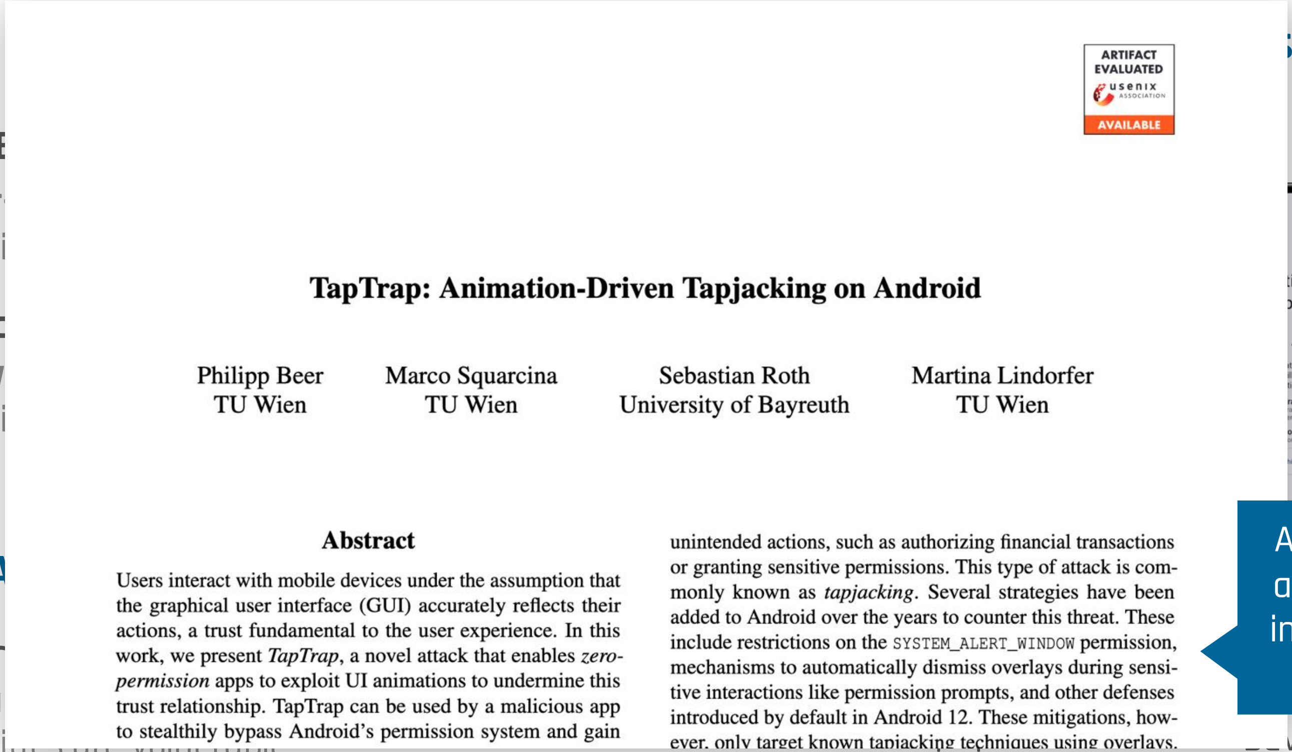
3rd Party A

Analysis of ~

76% of apps a
7% of all activities are vulnerable

s

permissions



ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

TapTrap: Animation-Driven Tapjacking on Android

Philipp Beer
TU Wien

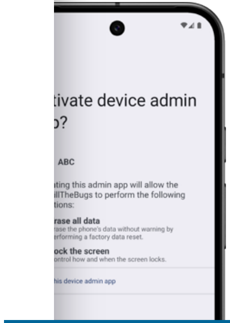
Marco Squarcina
TU Wien

Sebastian Roth
University of Bayreuth

Martina Lindorfer
TU Wien

Abstract

Users interact with mobile devices under the assumption that the graphical user interface (GUI) accurately reflects their actions, a trust fundamental to the user experience. In this work, we present *TapTrap*, a novel attack that enables *zero-permission* apps to exploit UI animations to undermine this trust relationship. TapTrap can be used by a malicious app to stealthily bypass Android's permission system and gain unintended actions, such as authorizing financial transactions or granting sensitive permissions. This type of attack is commonly known as *tapjacking*. Several strategies have been added to Android over the years to counter this threat. These include restrictions on the `SYSTEM_ALERT_WINDOW` permission, mechanisms to automatically dismiss overlays during sensitive interactions like permission prompts, and other defenses introduced by default in Android 12. These mitigations, however, only target known tapjacking techniques using overlays.



tivate device admin
p?

ABC

ting this admin app will allow the illTheBugs to perform the following tions:

rase all data
ase the phone's data without warning by rforming a factory data reset.

ock the screen
ontrol how and when the screen locks.

his device admin app

…vice erasure

Still vulnerable

Also includes an analysis of apps in the wild and a user study!

Thank You