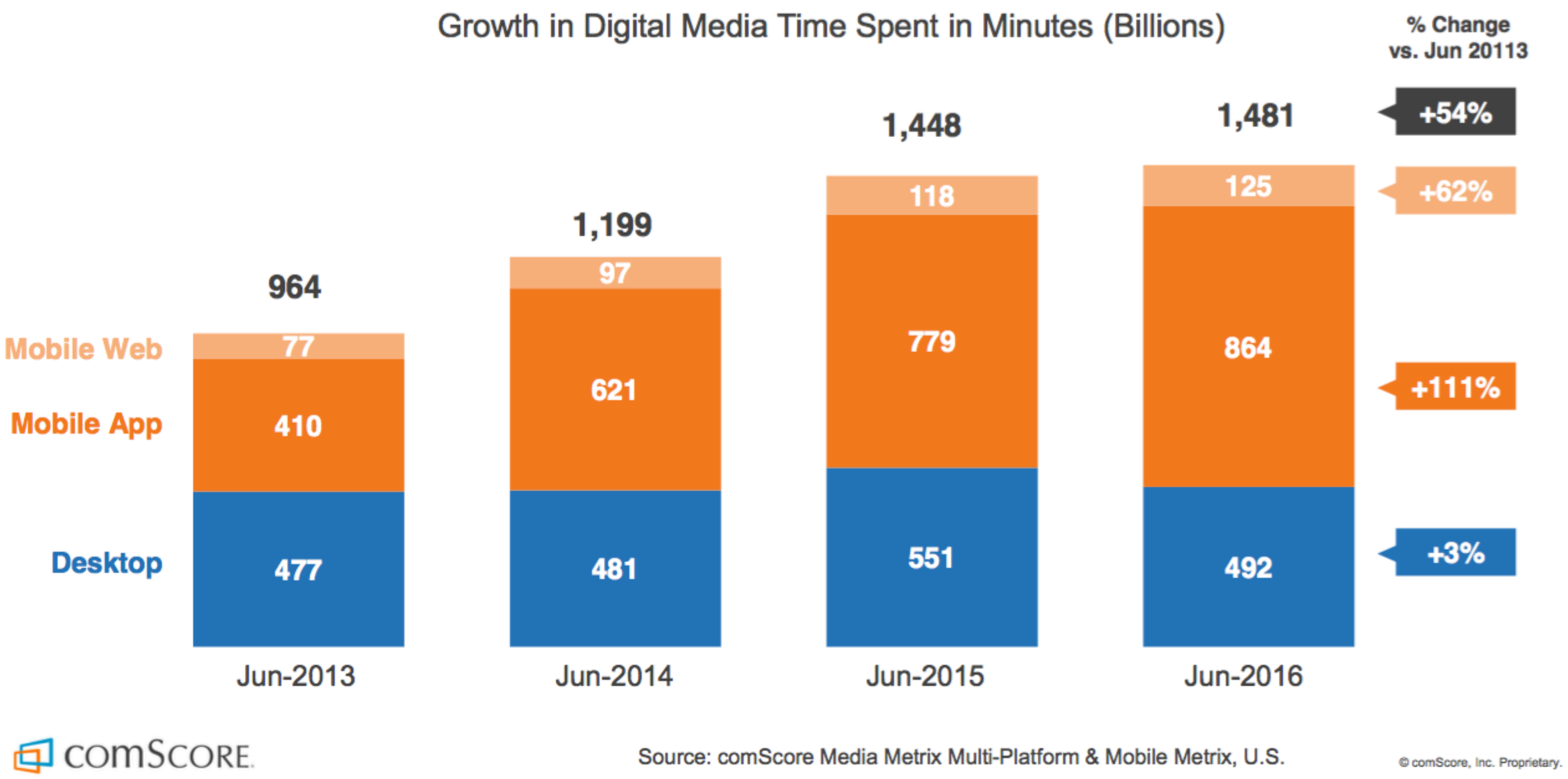
Mobile Security

CS155 Computer and Network Security

Stanford University

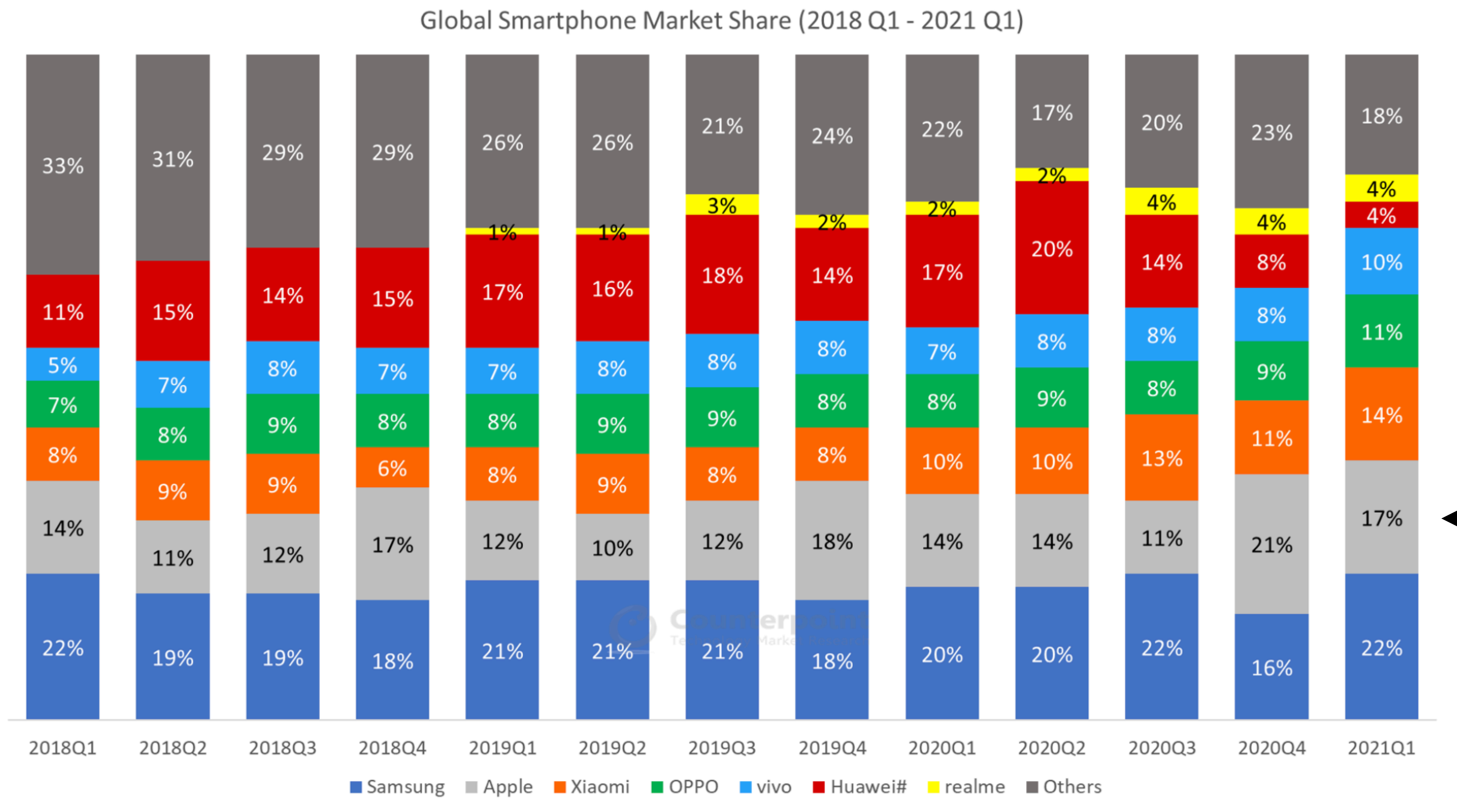
Mobile is Big!

3.8B mobile users worldwide. Users spend more time on mobile than on desktops today (exact numbers iffy)



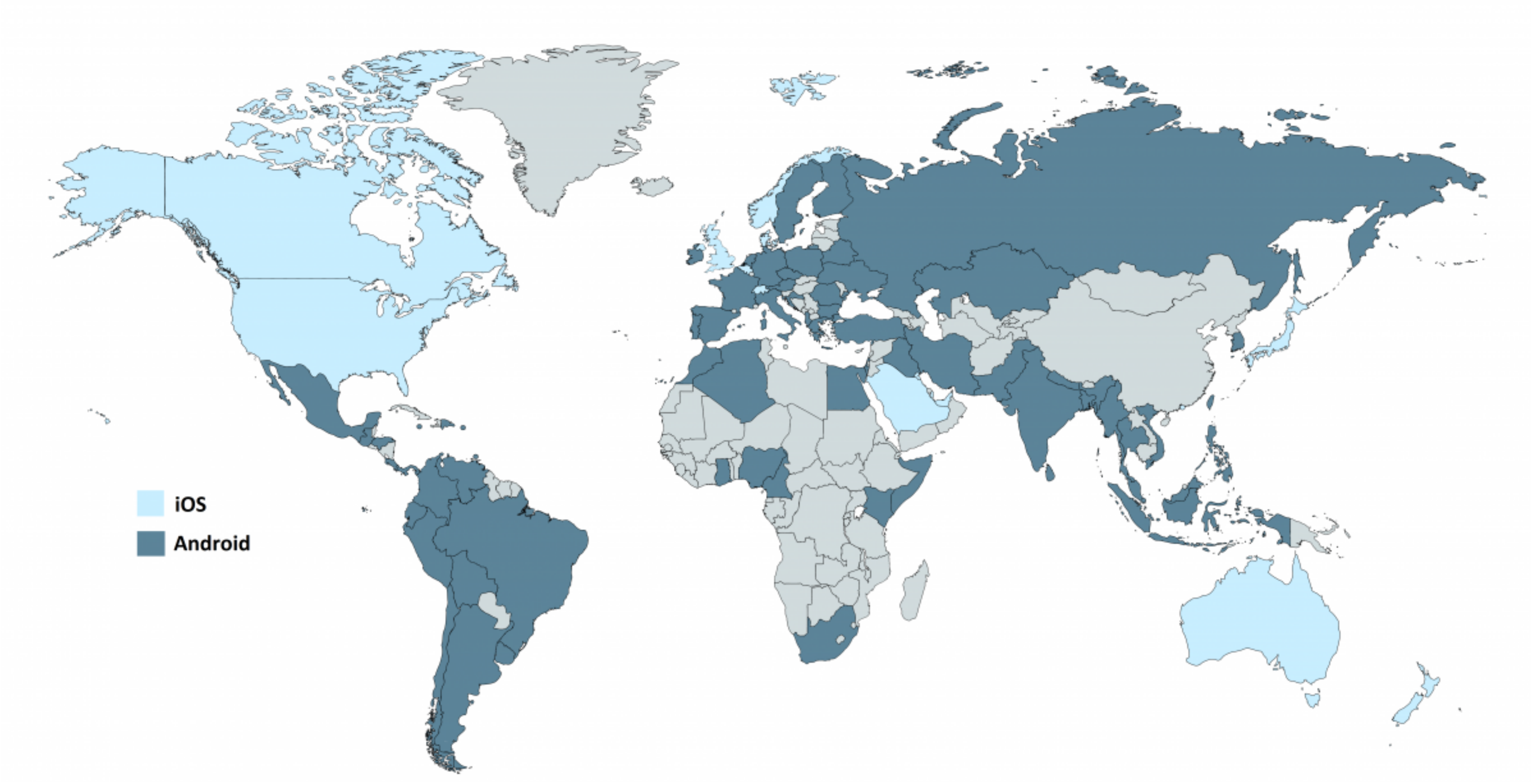
Source: comScore Media Metrix Multi-Platform & Mobile Metrix, U.S.

Mobile Market Share


Global Smartphone Market Share (2018 Q1 - 2021 Q1)

← 10-20% Apple.
Most Android.

Global Bias in Market Share



What’s Valuable on Phones?

Traditional (Similar to Desktop PCs)

- Steal data (e.g., contact list, email, messages, banking information, photos)
- Phishing
- Malvertising
- Join Bots

Mobile Specific

– Identify location

– Record phone calls

– Log SMS (What about 2FA SMS?)

– Send premium SMS messages

Unique Threat Model (Physical)

Powered-off devices under complete physical control of an adversary (including well-resourced nation states)

Screen locked devices under physical control of adversary (e.g. thieves)

Unlocked devices under control of different user (e.g. intimate partner abuse)

Devices in physical proximity to an adversary (with the assumed capability to control radio channels, including cellular, WiFi, Bluetooth, GPS, NFC)

Threat Model (Untrusted Code)

Android intentionally allows (with explicit consent by end users) installation of application code from arbitrary sources

Abusing APIs supported by the OS with malicious intent, e.g. spyware

Exploiting bugs in the OS, e.g. kernel, drivers, or system services

Mimicking system or other app user interfaces to confuse users

Reading content from system or other application user interfaces (e.g., screen-scrape)

Injecting input events into system or other app user interfaces

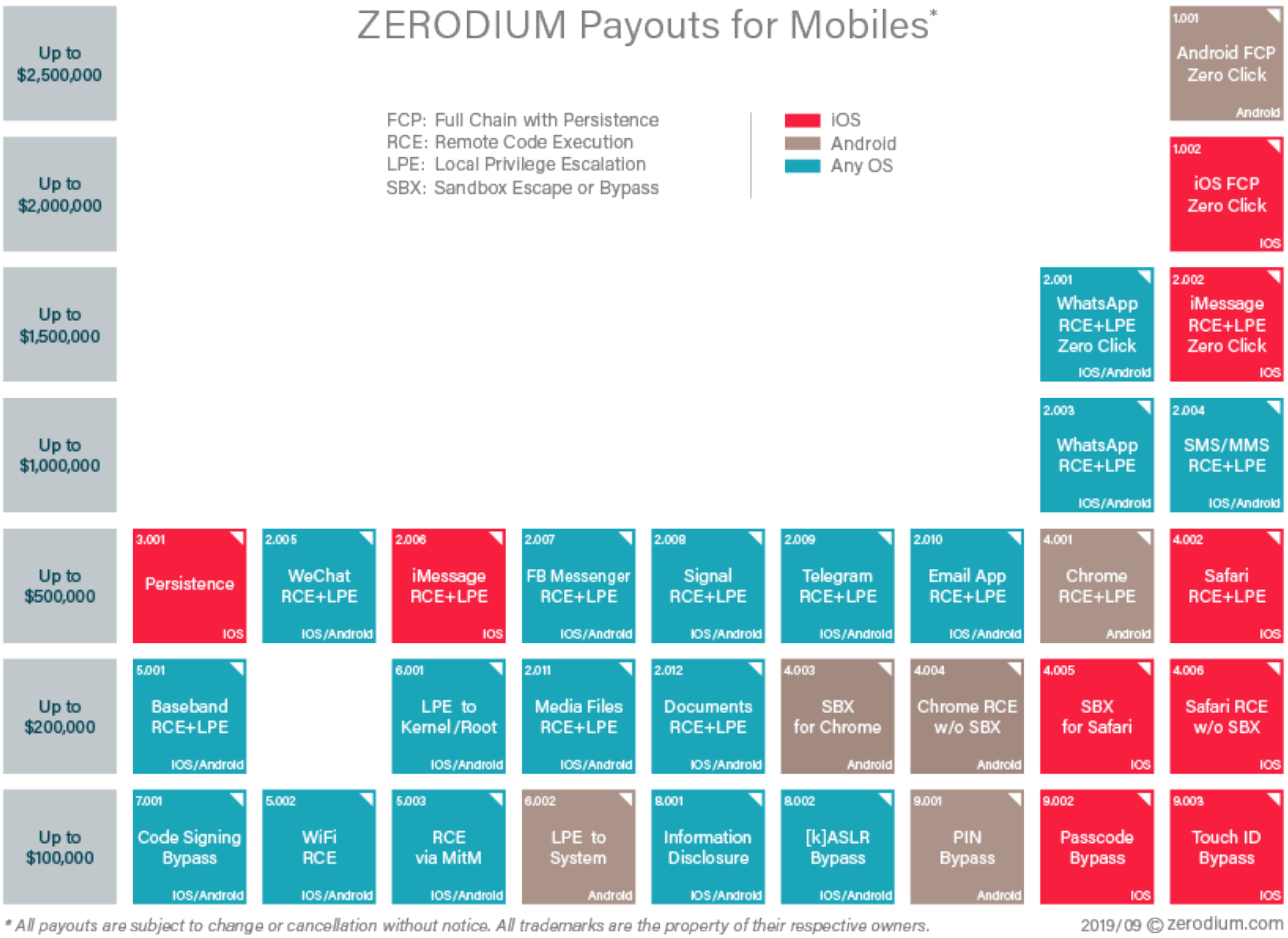
Unique Threat Model (Network)

The standard assumption of network communication under complete control of an adversary certainly also holds for Android. Assume fist hop (e.g., router) is also malicious.

Passive eavesdropping and traffic analysis, including tracking devices within or across networks (e.g. based on MAC address or other device network identifiers)

Active manipulation of network traffic (e.g. MITM on TLS)

Mobile Exploits Very Valuable



ZERODIUM Payouts for Mobiles*

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

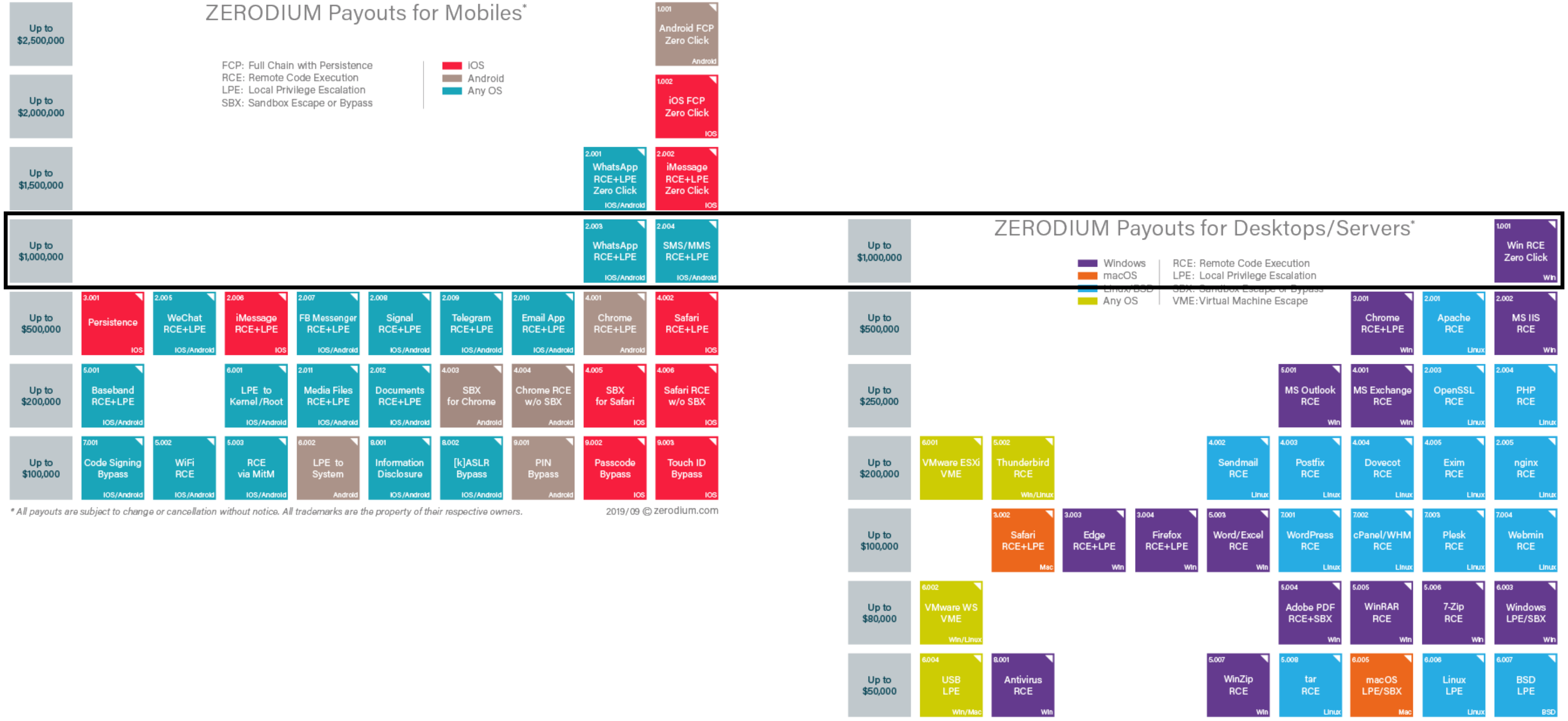
🟥 iOS
🟫 Android
🟦 Any OS

Up to $2,500,000									1.001 Android FCP Zero Click Android
Up to $2,000,000									1.002 iOS FCP Zero Click IOS
Up to $1,500,000								2.001 WhatsApp RCE+LPE Zero Click IOS/Android	2.002 iMessage RCE+LPE Zero Click IOS
Up to $1,000,000								2.003 WhatsApp RCE+LPE IOS/Android	2.004 SMS/MMS RCE+LPE IOS/Android
Up to $500,000	3.001 Persistence IOS	2.005 WeChat RCE+LPE IOS/Android	2.006 iMessage RCE+LPE IOS	2.007 FB Messenger RCE+LPE IOS/Android	2.008 Signal RCE+LPE IOS/Android	2.009 Telegram RCE+LPE IOS/Android	2.010 Email App RCE+LPE IOS/Android	4.001 Chrome RCE+LPE Android	4.002 Safari RCE+LPE IOS
Up to $200,000	5.001 Baseband RCE+LPE IOS/Android		6.001 LPE to Kernel/Root IOS/Android	2.011 Media Files RCE+LPE IOS/Android	2.012 Documents RCE+LPE IOS/Android	4.003 SBX for Chrome Android	4.004 Chrome RCE w/o SBX Android	4.005 SBX for Safari IOS	4.006 Safari RCE w/o SBX IOS
Up to $100,000	7.001 Code Signing Bypass IOS/Android	5.002 WiFi RCE IOS/Android	5.003 RCE via MitM IOS/Android	6.002 LPE to System Android	8.001 Information Disclosure IOS/Android	8.002 [k]ASLR Bypass IOS/Android	9.001 PIN Bypass Android	9.002 Passcode Bypass IOS	9.003 Touch ID Bypass IOS

* *All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.*

2019/09 © zerodium.com

Mobile Exploits Very Valuable



ZERODIUM Payouts for Mobiles*

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

🟥 iOS
🟫 Android
🟦 Any OS

Up to $2,500,000									1.001 Android FCP Zero Click Android
Up to $2,000,000									1.002 iOS FCP Zero Click iOS
Up to $1,500,000								2.001 WhatsApp RCE+LPE Zero Click iOS/Android	2.002 iMessage RCE+LPE Zero Click iOS
Up to $1,000,000								2.003 WhatsApp RCE+LPE iOS/Android	2.004 SMS/MMS RCE+LPE iOS/Android
Up to $500,000	3.001 Persistence iOS	2.005 WeChat RCE+LPE iOS/Android	2.006 iMessage RCE+LPE iOS	2.007 FB Messenger RCE+LPE iOS/Android	2.008 Signal RCE+LPE iOS/Android	2.009 Telegram RCE+LPE iOS/Android	2.010 Email App RCE+LPE iOS/Android	4.001 Chrome RCE+LPE Android	4.002 Safari RCE+LPE iOS
Up to $200,000	5.001 Baseband RCE+LPE iOS/Android		6.001 LPE to Kernel/Root iOS/Android	2.011 Media Files RCE+LPE iOS/Android	2.012 Documents RCE+LPE iOS/Android	4.003 SBX for Chrome Android	4.004 Chrome RCE w/o SBX Android	4.005 SBX for Safari iOS	4.006 Safari RCE w/o SBX iOS
Up to $100,000	7.001 Code Signing Bypass iOS/Android	5.002 WiFi RCE iOS/Android	5.003 RCE via MitM iOS/Android	6.002 LPE to System Android	8.001 Information Disclosure iOS/Android	8.002 [k]ASLR Bypass iOS/Android	9.001 PIN Bypass Android	9.002 Passcode Bypass iOS	9.003 Touch ID Bypass iOS

* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.

2019/09 © zerodium.com

ZERODIUM Payouts for Desktops/Servers*

🟪 Windows
🟧 macOS
🟦 Linux/BSD
🟨 Any OS

RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass
VME: Virtual Machine Escape

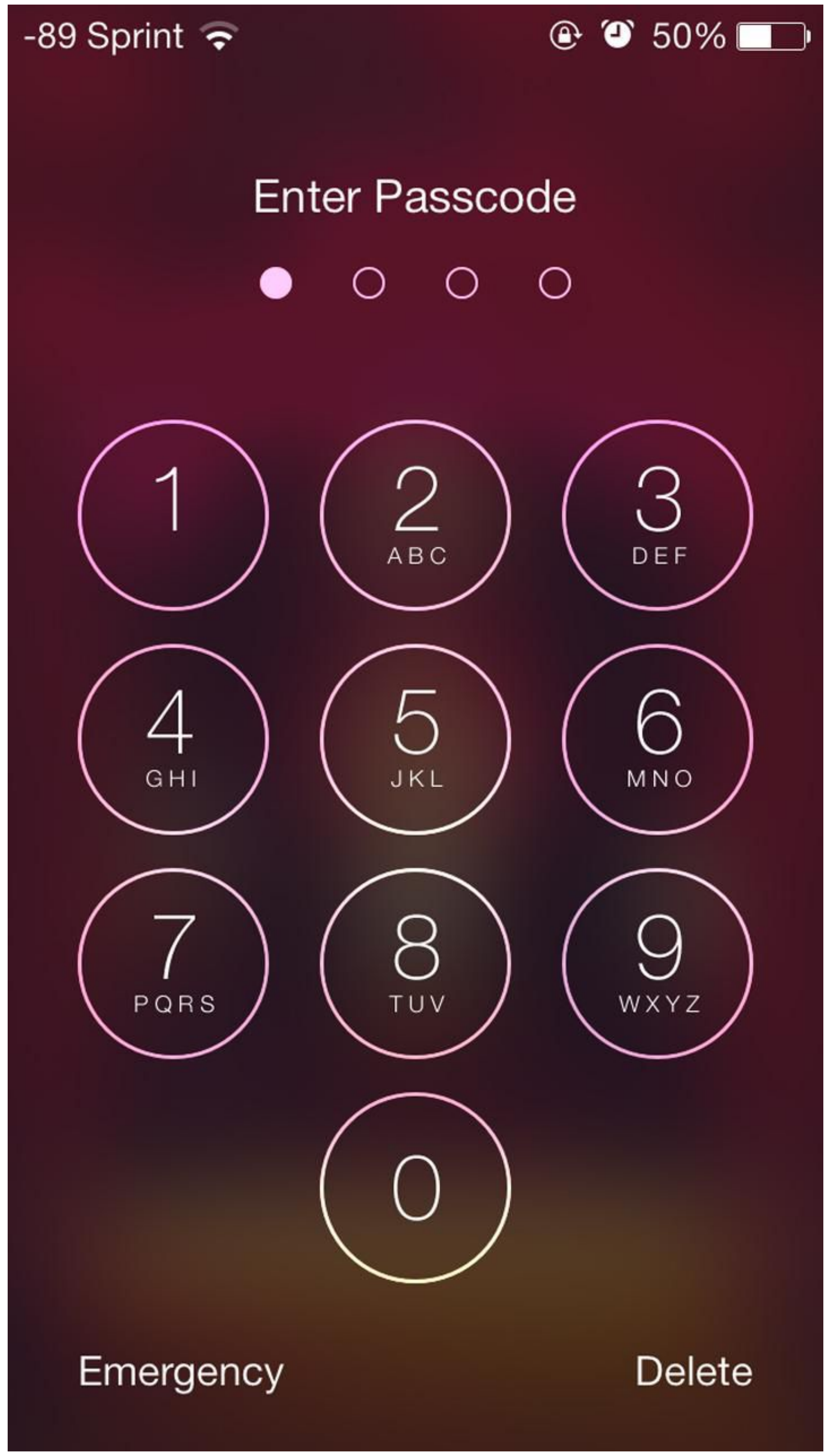
Up to $1,000,000									1.001 Win RCE Zero Click Win
Up to $500,000							3.001 Chrome RCE+LPE Win	2.001 Apache RCE Linux	2.002 MS IIS RCE Win
Up to $250,000						5.001 MS Outlook RCE Win	4.001 MS Exchange RCE Win	2.003 OpenSSL RCE Linux	2.004 PHP RCE Linux
Up to $200,000	6.001 VMware ESXi VME	5.002 Thunderbird RCE Win/Linux			4.002 Sendmail RCE Linux	4.003 Postfix RCE Linux	4.004 Dovecot RCE Linux	4.005 Exim RCE Linux	2.005 nginx RCE Linux
Up to $100,000		3.002 Safari RCE+LPE Mac	3.003 Edge RCE+LPE Win	3.004 Firefox RCE+LPE Win	5.003 Word/Excel RCE Win	7.001 WordPress RCE Linux	7.002 cPanel/WHM RCE Linux	7.003 Plesk RCE Linux	7.004 Webmin RCE Linux
Up to $80,000	6.002 VMware WS VME Win/Linux					5.004 Adobe PDF RCE+SBX Win	5.005 WinRAR RCE Win	5.006 7-Zip RCE Win	6.003 Windows LPE/SBX Win
Up to $50,000	6.004 USB LPE Win/Mac	8.001 Antivirus RCE Win			5.007 WinZip RCE Win	5.008 tar RCE Linux	6.005 macOS LPE/SBX Mac	6.006 Linux LPE Linux	6.007 BSD LPE BSD

Physical Security

Unlocking Device

Typically: Need PIN, pattern, or alphanumeric password to unlock device

Some applications (e.g., banking apps) also require entering a PIN to access the app



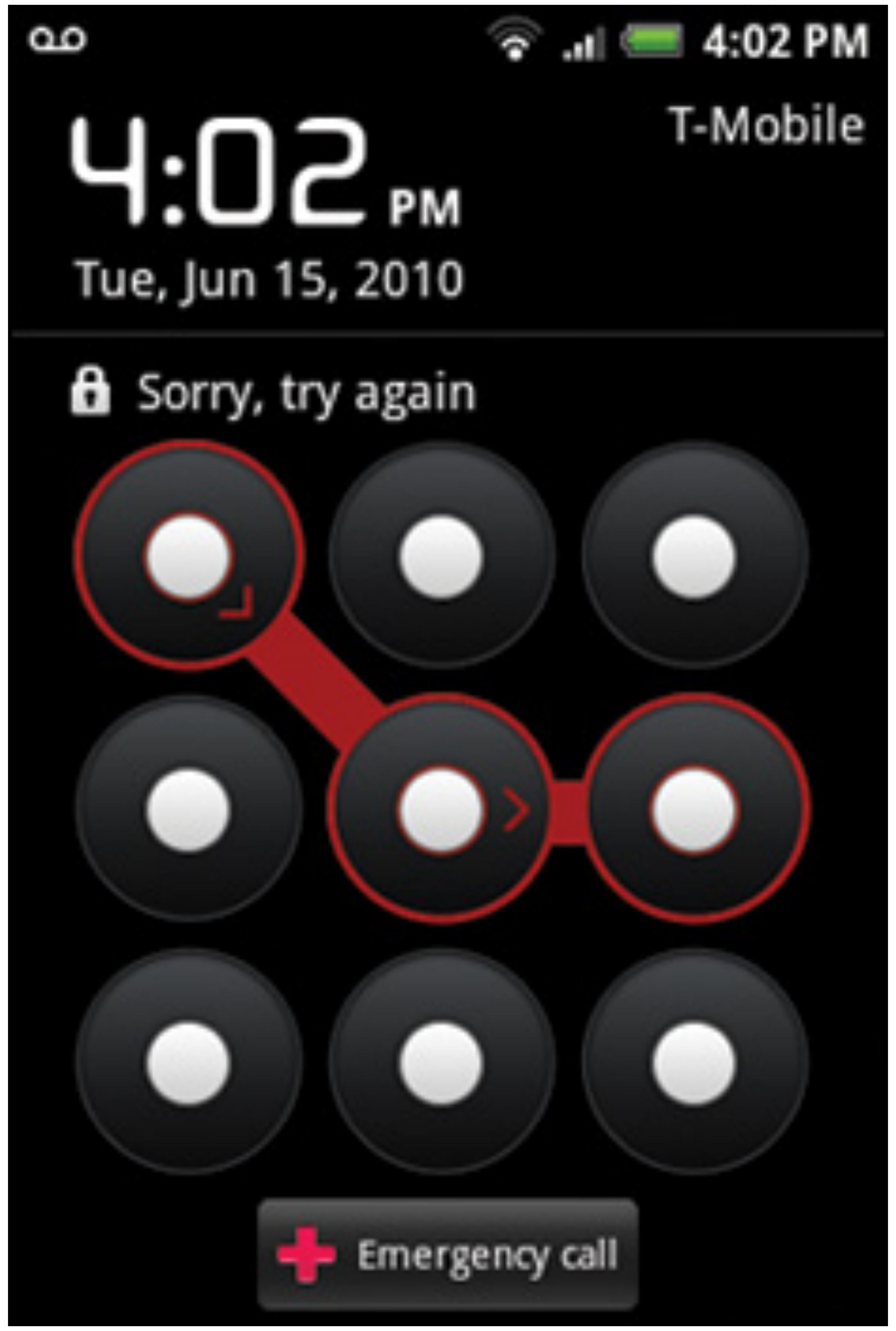
Swipe Code Problems

Smudge attacks [Aviv et al., 2010]

- Entering pattern leaves smudge that can be detected with proper lighting
- Smudge survives incidental contact with clothing

Another problem: entropy

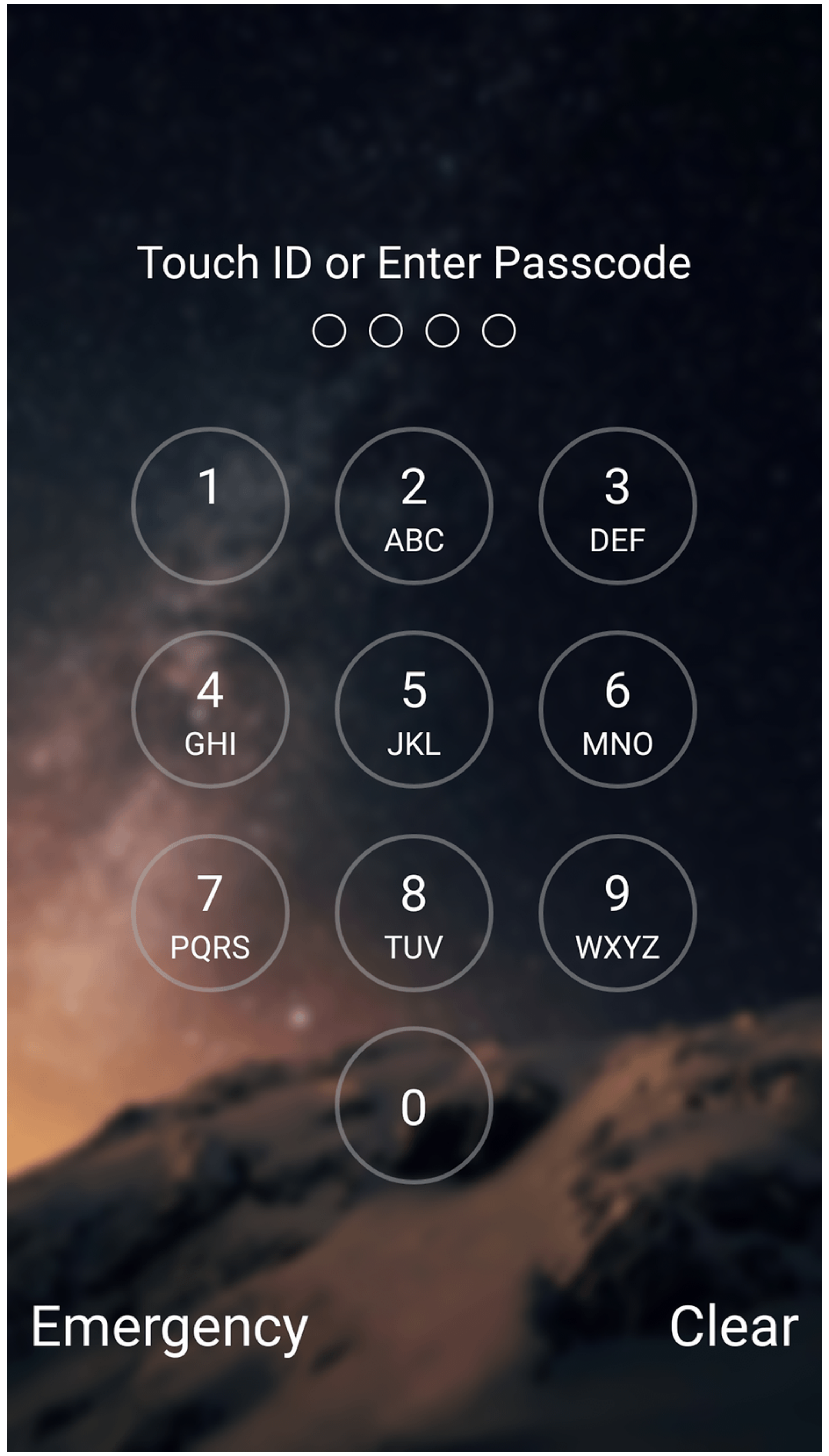
- People choose simple patterns – few strokes
- At most 1,600 patterns with <5 strokes



Sorry, try again

Passcodes

How do you allow a 4-6 digit PIN and be secure?



Review: Modern Password Hashing

Store Salted Hash (Best)

- Store (r, **H**(pw || r)) and check match against **H**(input || r)
- Prevents attackers from pre-computing password hashes

Making sure to choose an **H** that's expensive to compute:

SHA-512: 3,235 MH/s
SHA-3 (Keccak): 2,500 MH/s
BCrypt: 43,551 H/s

Use one of bcrypt, scrypt, or pbkdf2 when building an application

iPhone Password Hashing

Come up password hashing approach where 4-6 digits takes a very long time to crack, even if the device is physically compromised…

Additional Constraints:

- Lots of computation uses up battery (limited resource)!
- Physical access allows copying secret off and cracking remotely

Secure Enclave

iPhones have a second secure processor known as "secure enclave"

- Memory is inaccessible to normal OS
- Secure boot process that ensures its software is signed
- Each secure enclave has an AES key burned in at manufacture.

Processor has instructions that allow encrypting and decrypting content using the stored key, but the key itself is never accessible (incl. via JTAG)

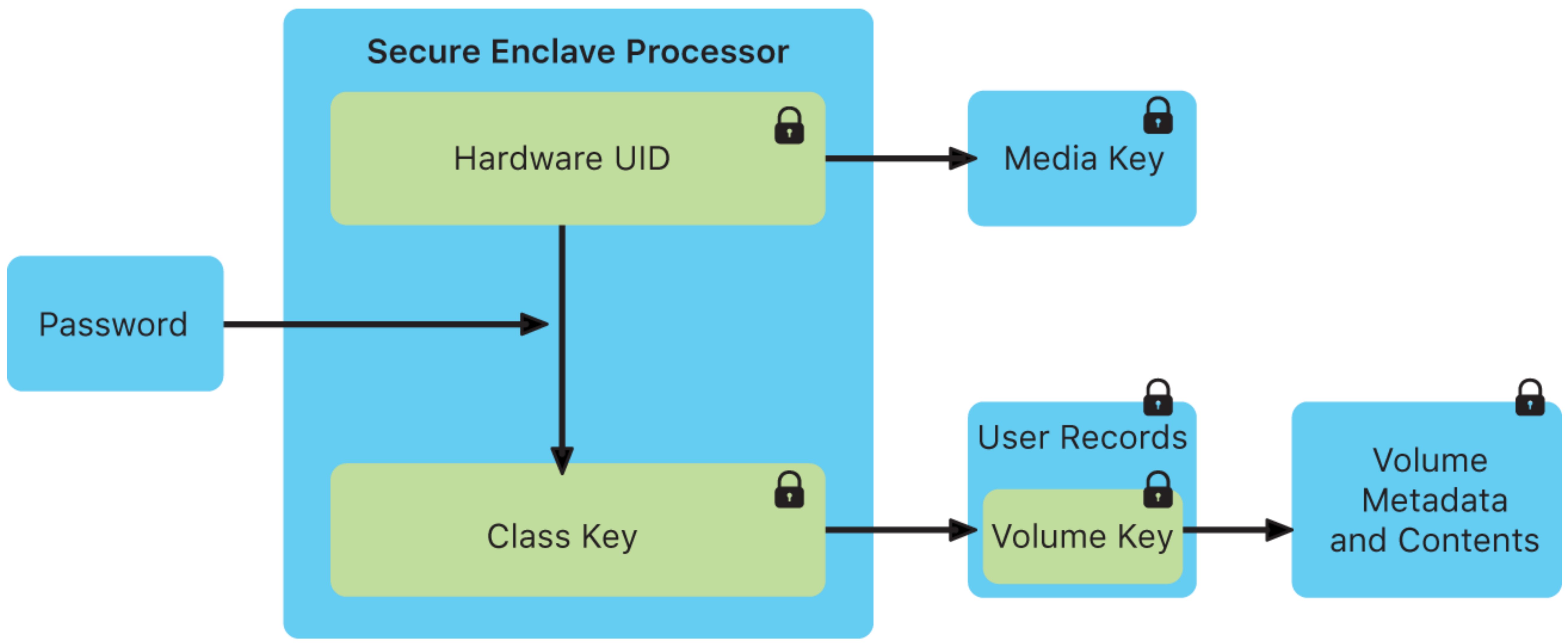
iPhone Unlocking

User passcode is intertwined with AES key fused into secure enclave (known as UID) when it is entered by the user

Imagine: key = $\text{Encrypt}_{\text{UID}}$(passcode).

This means that the the key to decrypt the device can only be derived on the single secure enclave on a specific phone. Not possible to take offline and brute force.

iPhone Unlocking Key



What prevents asking secure enclave repeatedly to try different passwords?

The passcode is entangled with the device’s UID many times —requires approximately 80ms per password guess.

Imagine: $\text{Encrypt}_{\text{UID}}(\text{Encrypt}_{\text{UID}}(\text{Encrypt}_{\text{UID}}(\text{passcode})\ldots))$

iPhone Unlock Time Estimate

At 80ms per password check…

- 5.5 years to try all 6 digits pins
- 5 failed attempts ⇒ 1min delay, 9 failures ⇒ 1 hour delay
  - >10 failed attempts ⇒ erase phone

All of this enforced by firmware on the secure enclave itself — cannot be changed by any malware that controls iOS

FBI–Apple Encryption Dispute

After the San Bernardino shooting in 2016, FBI tried to compel Apple to “unlock” iPhone. What were they specifically requesting?

Not possible to make password guessing any faster—innately dependent on performance of burned-in AES key

FBI–Apple Encryption Dispute

Remember…

- 5 failed attempts ⇒ 1min delay, 9 failures ⇒ 1 hour delay
  - >10 failed attempts ⇒ erase phone

This is managed by code on the secure enclave, which *can* be updated by Apple, not managed in hardware.

Technical Details

The court order wanted a custom version of a secure enclave firmware that would…

1. "it will bypass or disable the auto-erase function whether or not it has been enabled" (this user-configurable feature of iOS 8 automatically deletes keys needed to read encrypted data after ten consecutive incorrect attempts)

2. "it will enable the FBI to submit passcodes to the SUBJECT DEVICE for testing electronically via the physical device port, Bluetooth, Wi-Fi, or other protocol"

3. "it will ensure that when the FBI submits passcodes to the SUBJECT DEVICE, software running on the device will not purposefully introduce any additional delay between passcode attempts beyond what is incurred by Apple hardware”

What happened?

Apple planned to fight the order, “*The United States government has demanded that Apple take an unprecedented step which threatens the security of our customers. We oppose this order, which has implications far beyond the legal case at hand. This moment calls for public discussion, and we want our customers and people around the country to understand what is at stake.*”

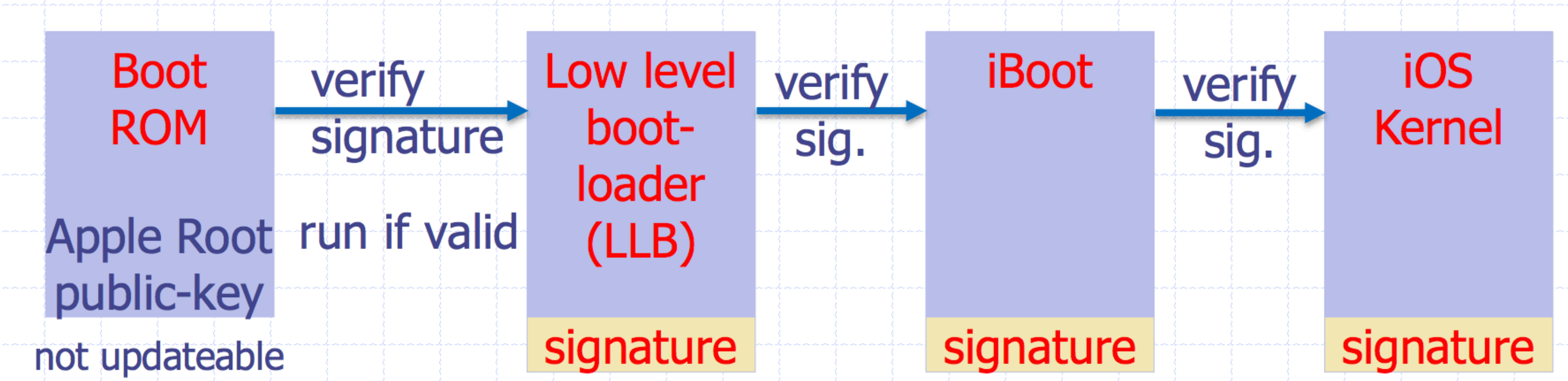
One day before hearing, FBI dropped the request, saying a third party had demonstrated a possible way to unlock the iPhone in question. No precent set re *all writs* act.

Secure Boot Chain

Why couldn't the FBI just upload their own firmware onto the secure enclave?

When an iOS device is turned on, it executes code from read-only memory known as Boot ROM. This immutable code, known as the hardware root of trust, is laid down during chip fabrication, and is implicitly trusted.

The Boot ROM code contains the Apple Root CA public key, which is used to verify that the bootloader is signed by Apple. This is the first step in the chain of trust where each step ensures that the next is signed by Apple.



Software Updates

To prevent devices from being *downgraded* to older versions that lack the security updates, iOS uses *System Software Authorization*.

Device connects to Apple with cryptographic descriptors of each component update (e.g., boot loader, kernel, and OS image), current versions, a random nonce, and device specific Exclusive Chip ID (ECID).

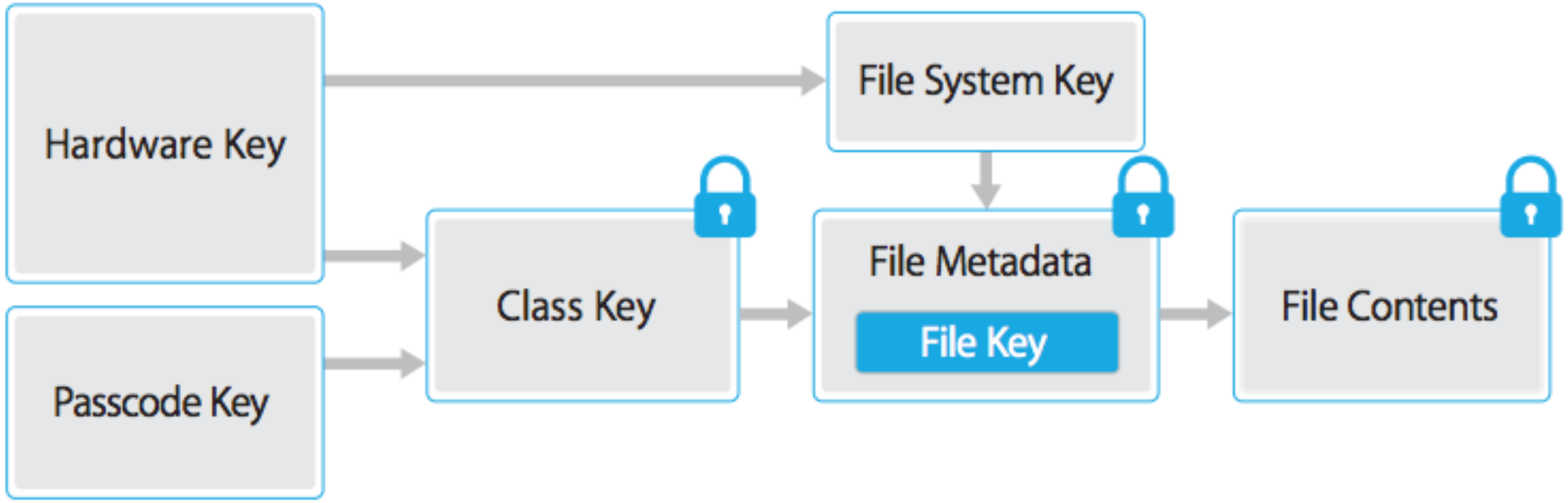
Apple signs device-personalized message allowing update, which boot loader verifies.

FaceID/TouchID

Files are encrypted through a hierarchy of encryption keys

Application files written to Flash are encrypted:

- Per-file key: encrypts all file contents (AES-XTS)
- Class key: encrypts per-file key (ciphertext stored in metadata)
- File-system key: encrypts file metadata



FaceID/TouchID

Files are encrypted through a hierarchy of encryption keys

By default (no FaceID, TouchID), class encryption keys are erased from memory of secure enclave whenever the device is locked or powered off

When TouchID/FaceID is enabled, class keys are kept and hardware sensor sends fingerprint image to secure enclave. All ML/analysis is performed within the secure enclave.

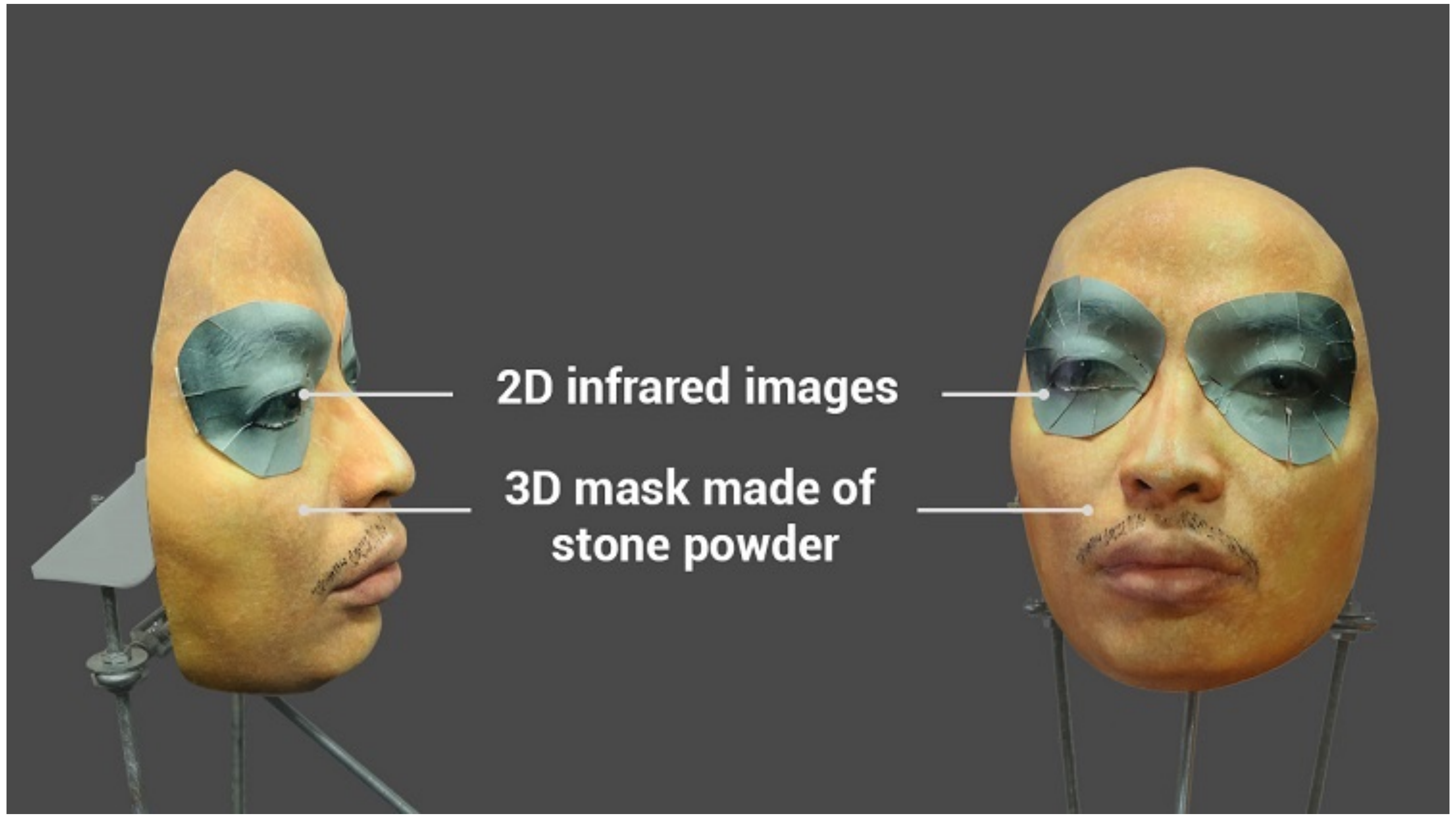
How Secure is TouchID?

Easy to build a fake finger if you have someone's fingerprint

- Several demos on YouTube. ~20 min
- Similar work on FaceID

The problem: fingerprints are not secret. Cannot replace.

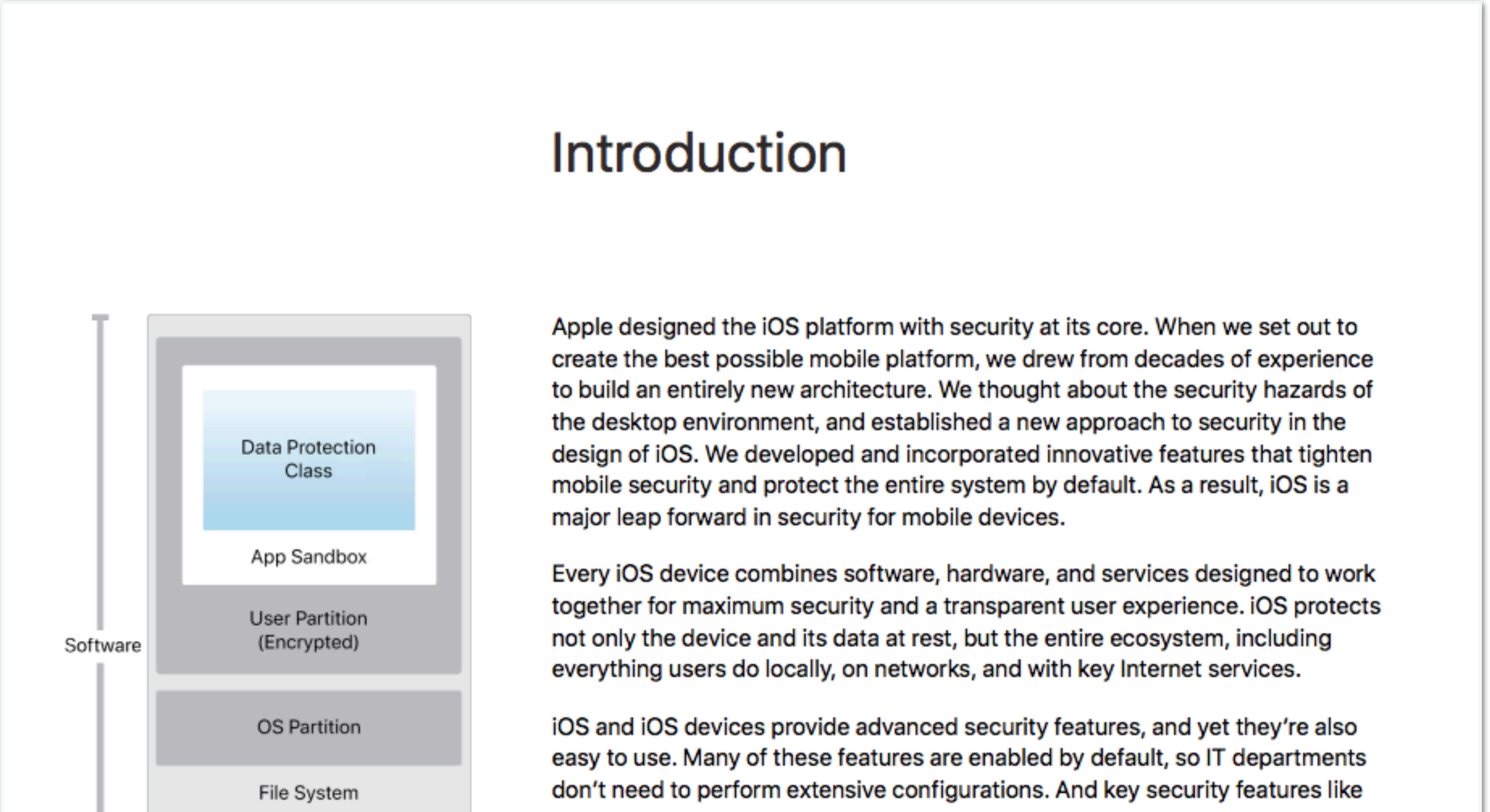
Convenient, but more secure solutions exist, e.g., unlock phone via bluetooth using a wearable device



More Information

iOS Security

https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf



Introduction

Apple designed the iOS platform with security at its core. When we set out to create the best possible mobile platform, we drew from decades of experience to build an entirely new architecture. We thought about the security hazards of the desktop environment, and established a new approach to security in the design of iOS. We developed and incorporated innovative features that tighten mobile security and protect the entire system by default. As a result, iOS is a major leap forward in security for mobile devices.

Every iOS device combines software, hardware, and services designed to work together for maximum security and a transparent user experience. iOS protects not only the device and its data at rest, but the entire ecosystem, including everything users do locally, on networks, and with key Internet services.

iOS and iOS devices provide advanced security features, and yet they're also easy to use. Many of these features are enabled by default, so IT departments don't need to perform extensive configurations. And key security features like

Bring Your Own Device (BYOD)

Many companies are now allowing users to bring/use their own personal devices — company data resides on devices

In the past, enterprise workstations were centrally managed.

How do you handle when users want to bring their own devices?

Mobile Device Management

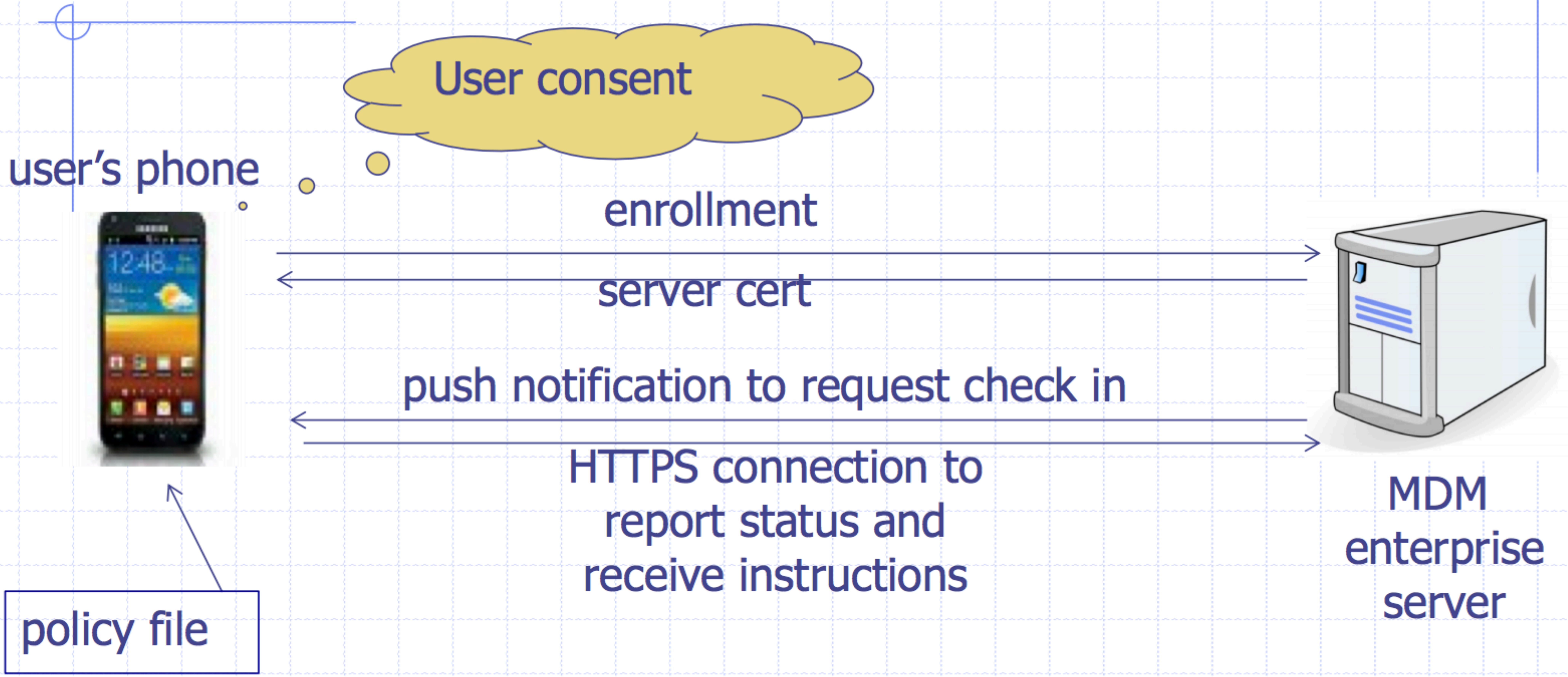
Manage mobile devices across organization

Consists of central server and client-side software. Now part of mobile OSes too.

Allows:

- Diagnostics, repair, and update
- Backup and restore
- Policy enforcement (e.g. only allowed apps)
- Remote lock and wipe
- GPS Tracking

Sample MDM Enrollment



User consent

user's phone

enrollment

server cert

push notification to request check in

HTTPS connection to
report status and
receive instructions

MDM
enterprise
server

policy file

configure, query, lock, wipe, …

Mobile Malware

What’s Different?

Applications are isolated

- Each runs in a separate execution context
- No default access to file system, devices, etc.
- Different than traditional OSes where multiple applications run with the same user permissions!

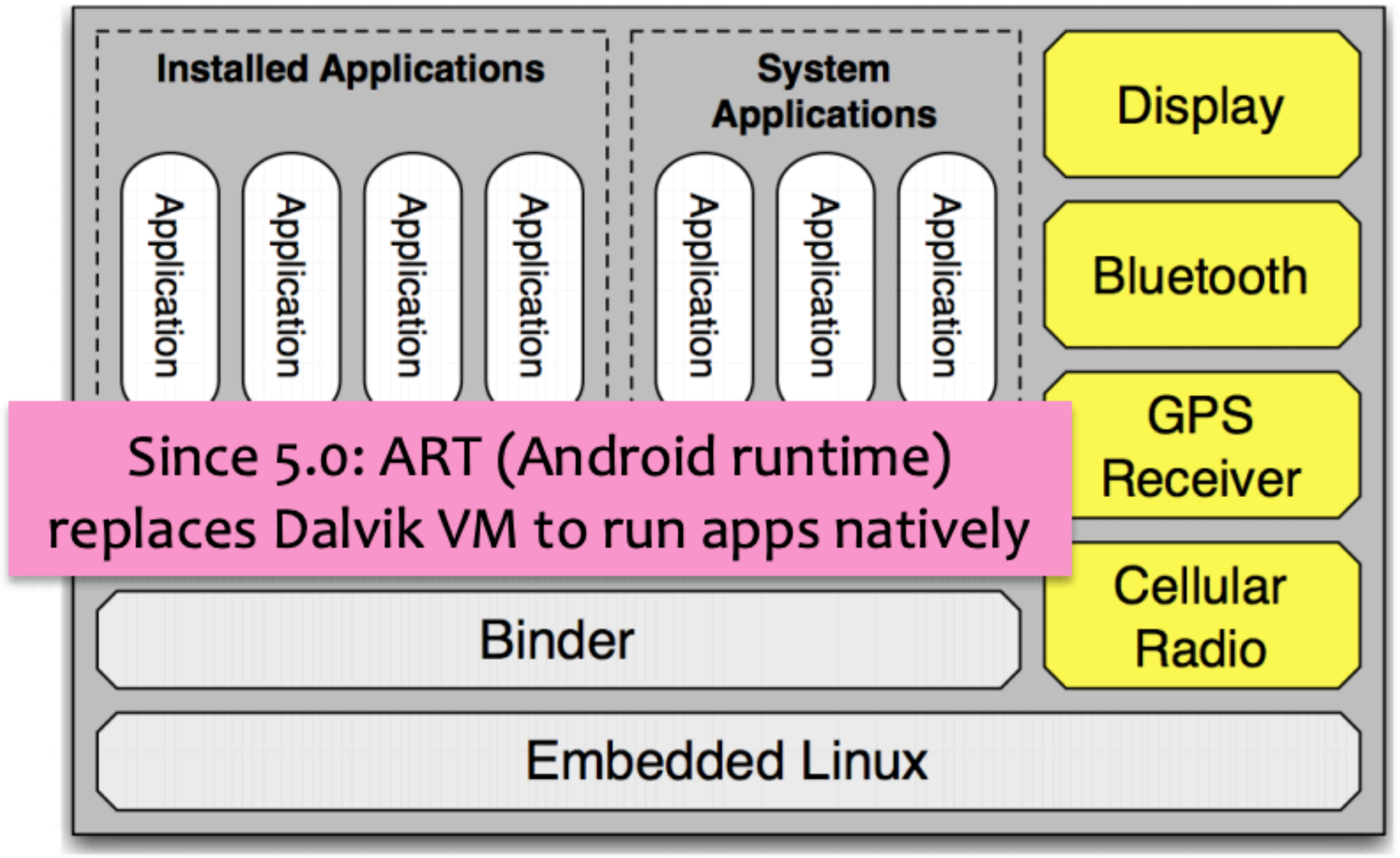
Applications are installed via App Store (and malware spreads)

- Market: Vendor controlled (Apple) / open (Android)
- User approval of permissions

Android Isolation

Based on Linux with sandboxes (SE Linux)

- Appls run as separate UIDs, in separate processes.
- Memory corruption errors only lead to arbitrary code execution in application, not complete system compromise!
- Can still escape sandbox – must compromise Linux kernel



Since 5.0: ART (Android runtime) replaces Dalvik VM to run apps natively

What is Rooting?

Allows user to run applications with root privileges, e.g., modify/delete system files and app, CPU, network management

Done by exploiting vulnerability in firmware to install a custom OS or firmware image

Double-edged sword… lots of malware only affects rooted devices

Examples of Malware

DroidDream (Android)

- Over 58 apps uploaded to Google app market
- Conducts data theft; send credentials to attackers

Attacked vulnerability in Android itself

Zitmo (Symbian, BlackBerry, Windows, Android)

- Poses as mobile banking application
- Captures info from SMS – steal banking 2FA codes
- Works with Zeus botnet

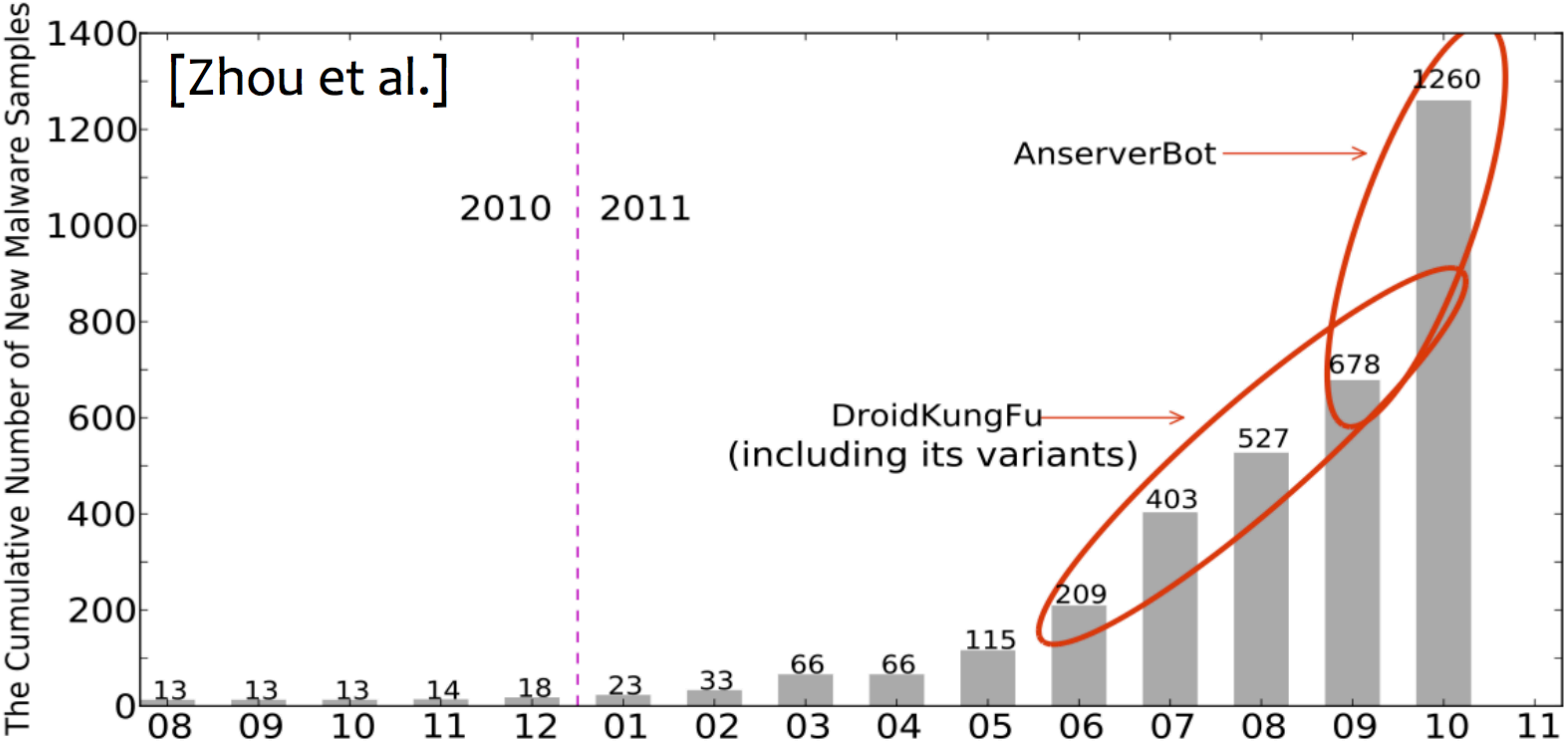
Malicious application that tricked users

Ikee (iOS)

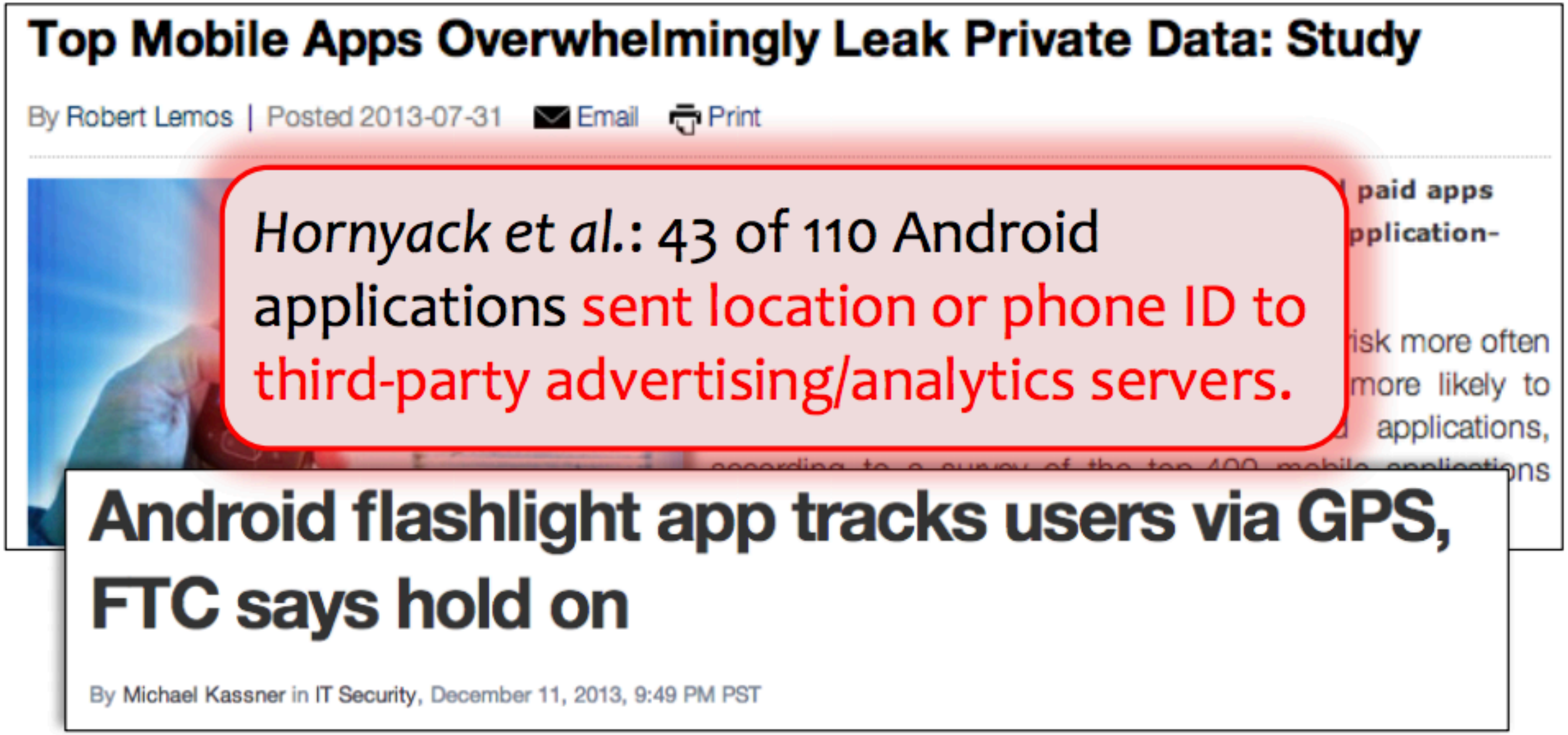
- Worm capabilities (targeted default ssh password)
- Worked only on jailbroken phones with ssh installed

Attacked vulnerability in rooted iPhones

Large Target for Attackers



Legitimate Apps Too…



Top Mobile Apps Overwhelmingly Leak Private Data: Study

By Robert Lemos | Posted 2013-07-31 Email Print

Hornyack et al.: 43 of 110 Android applications sent location or phone ID to third-party advertising/analytics servers.

paid apps
pplication-

isk more often
more likely to
applications,
according to a survey of the top 400 mobile applications

Android flashlight app tracks users via GPS, FTC says hold on

By Michael Kassner in IT Security, December 11, 2013, 9:49 PM PST

Challenges with Isolated Apps

So mobile platforms isolate applications for security, but….

1) **Permissions:** How can applications access sensitive resources?

2) **Communication:** How can applications communicate with each other?

Mobile Permissions

(1) Permission Granting Problem

Smartphones (and other modern OSes) try to prevent such attacks by limiting applications' default access to:

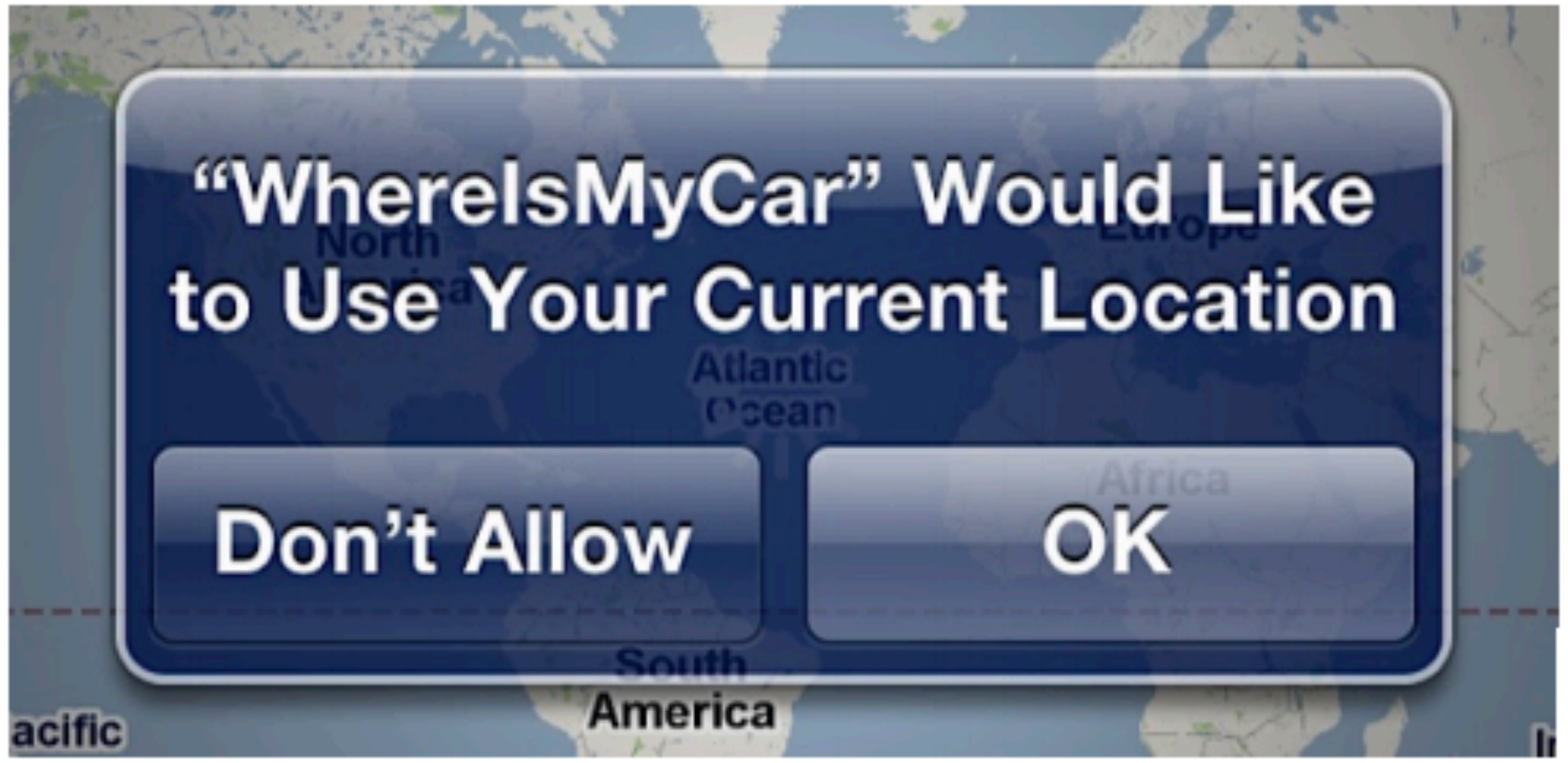
- System Resources (clipboard, file system)
- Devices (e.g., camera, GPS, phone, …)

How should operating system grant permissions to applications?

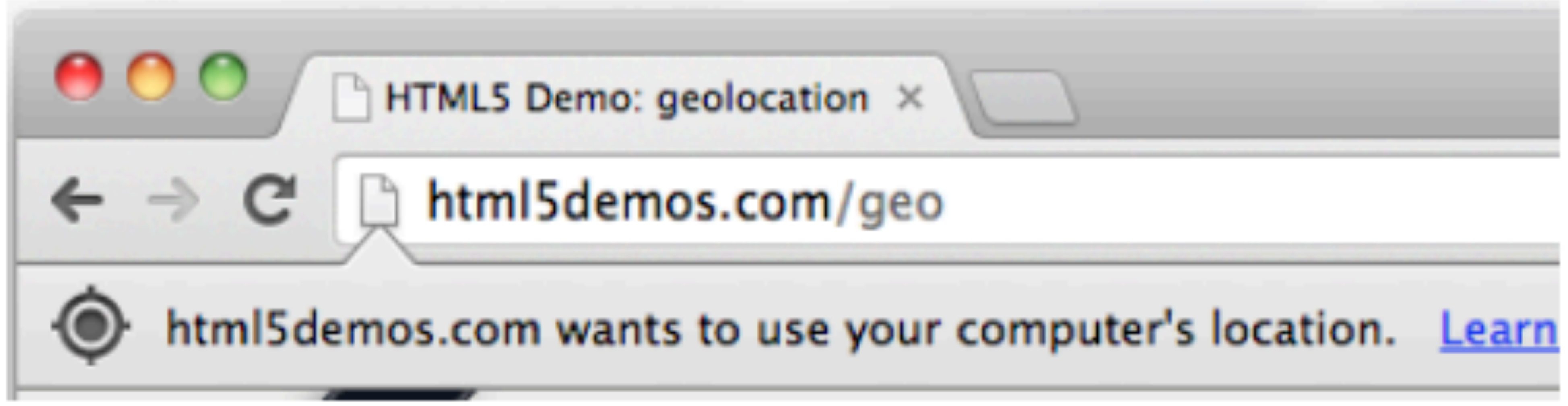
Standard approach: Ask the user.

State of the Art

Prompts (time-of-use)



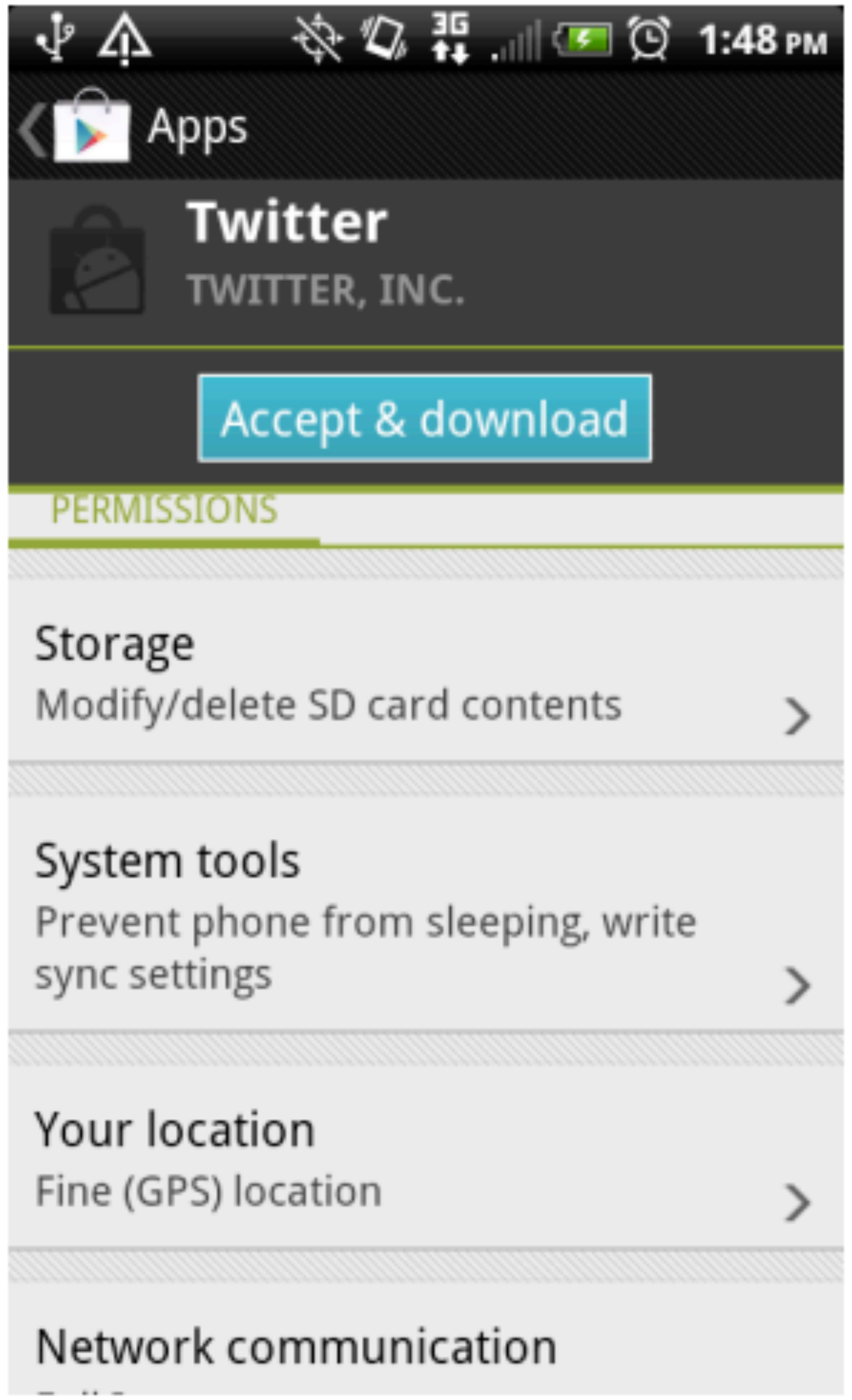
“WhereIsMyCar” Would Like to Use Your Current Location



html5demos.com/geo

html5demos.com wants to use your computer's location. Learn

Manifests (install-time)



Twitter

TWITTER, INC.

Storage
Modify/delete SD card contents

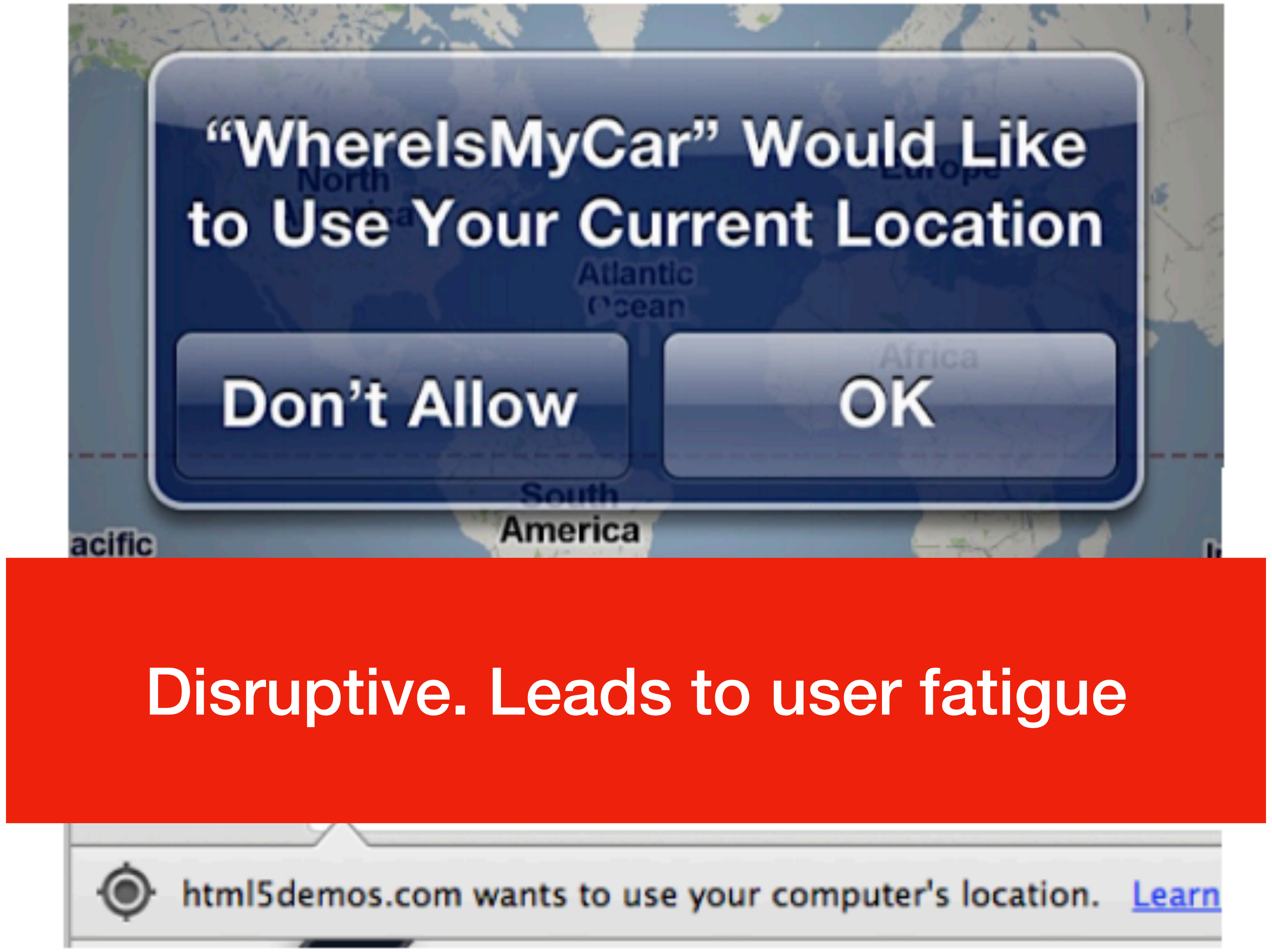
System tools
Prevent phone from sleeping, write sync settings

Your location
Fine (GPS) location

Network communication

State of the Art

Prompts (time-of-use)

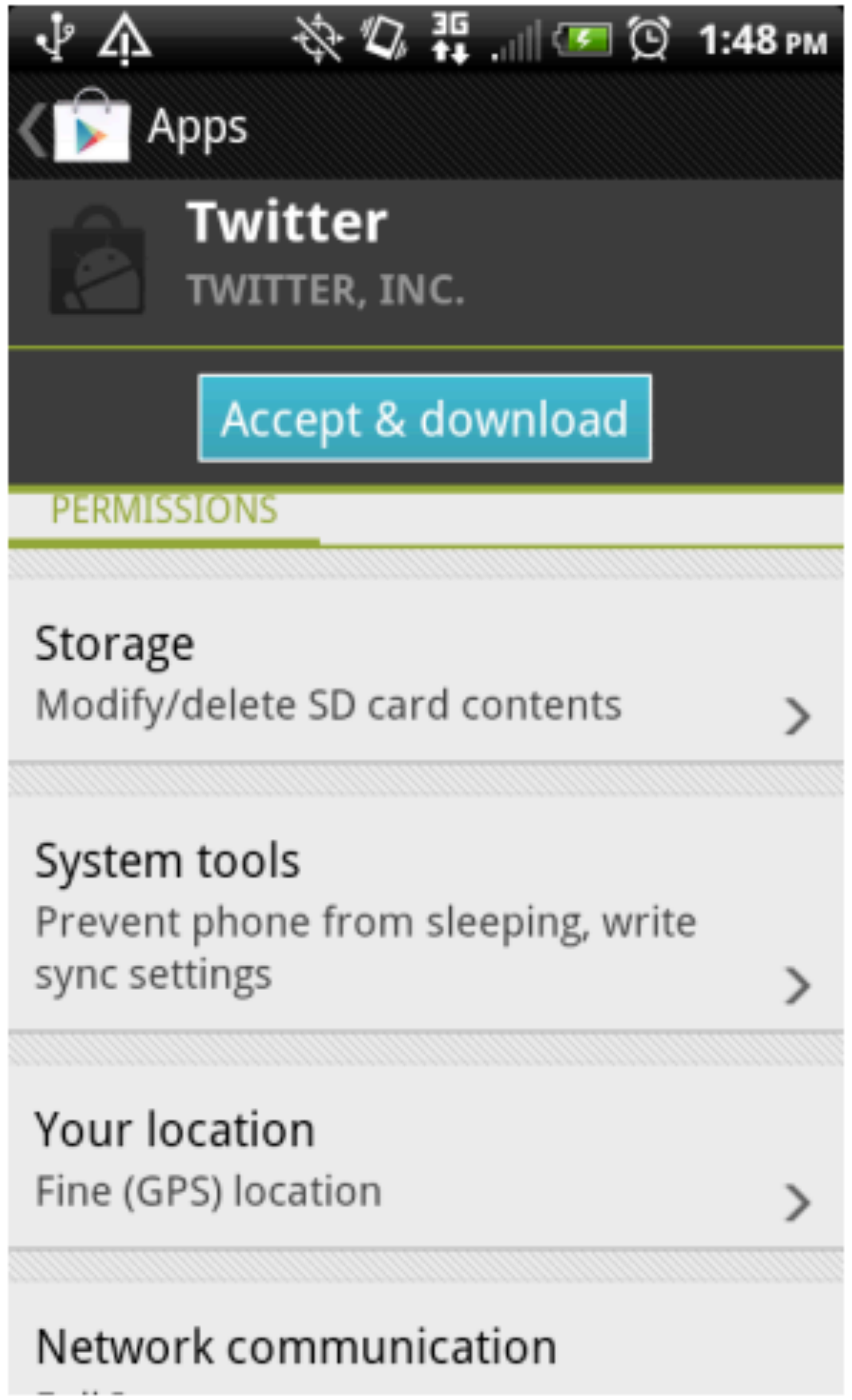


“WhereIsMyCar” Would Like to Use Your Current Location

html5demos.com wants to use your computer's location. Learn

Disruptive. Leads to user fatigue

Manifests (install-time)



Twitter

TWITTER, INC.

PERMISSIONS

Storage
Modify/delete SD card contents

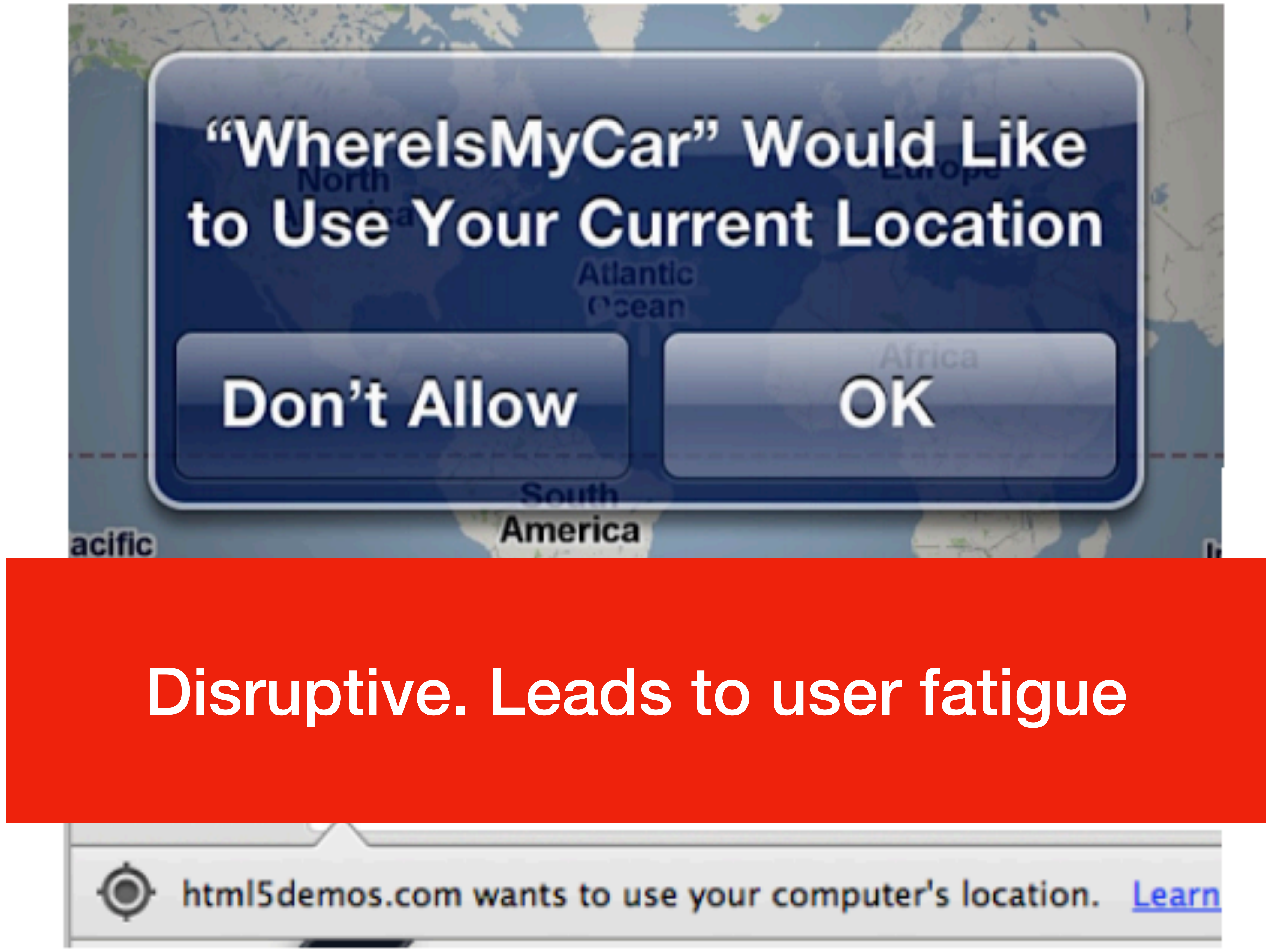
System tools
Prevent phone from sleeping, write sync settings

Your location
Fine (GPS) location

Network communication

State of the Art

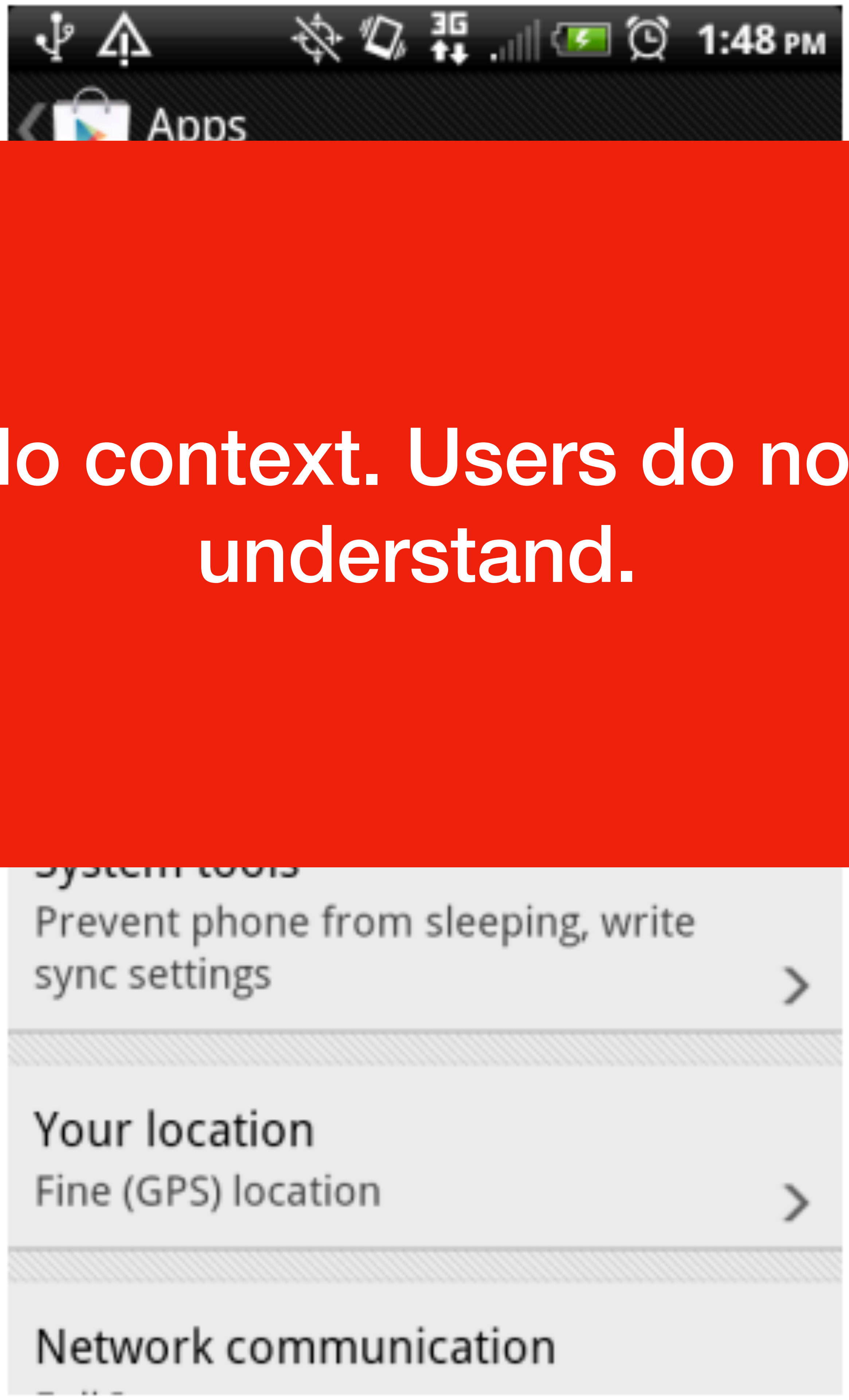
Prompts (time-of-use)


“WhereIsMyCar” Would Like to Use Your Current Location

Disruptive. Leads to user fatigue

html5demos.com wants to use your computer's location. Learn

Manifests (install-time)



No context. Users do not understand.

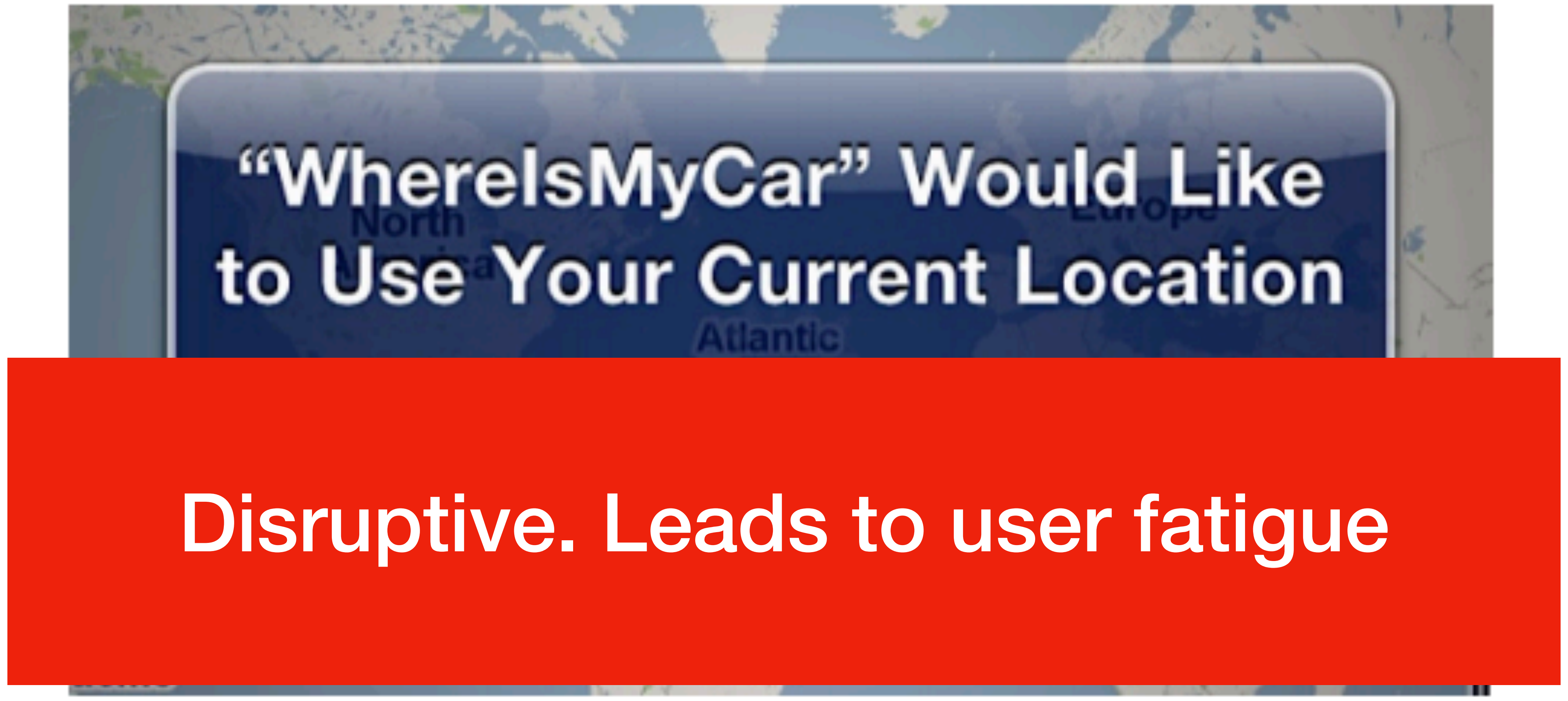
System tools
Prevent phone from sleeping, write sync settings

Your location
Fine (GPS) location

Network communication

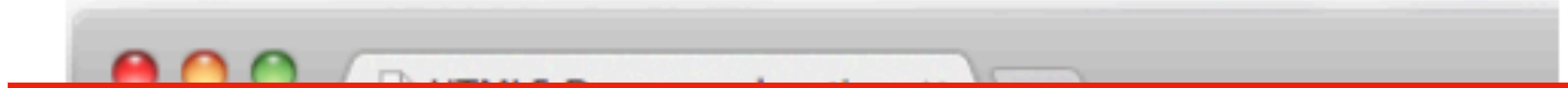
State of the Art

Prompts (time-of-use)

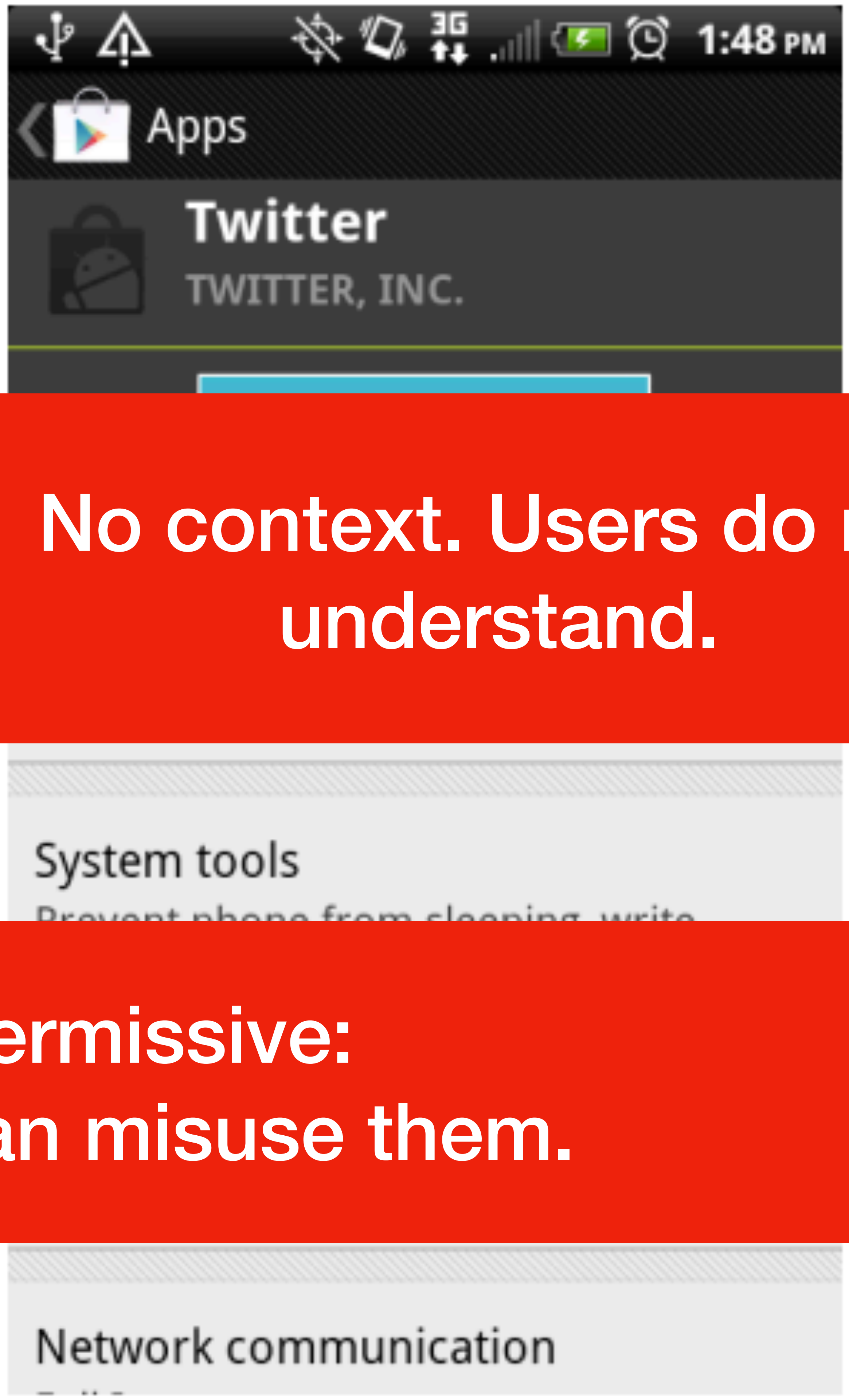


“WhereIsMyCar” Would Like to Use Your Current Location

Disruptive. Leads to user fatigue



Manifests (install-time)



Twitter

TWITTER, INC.

No context. Users do not understand.

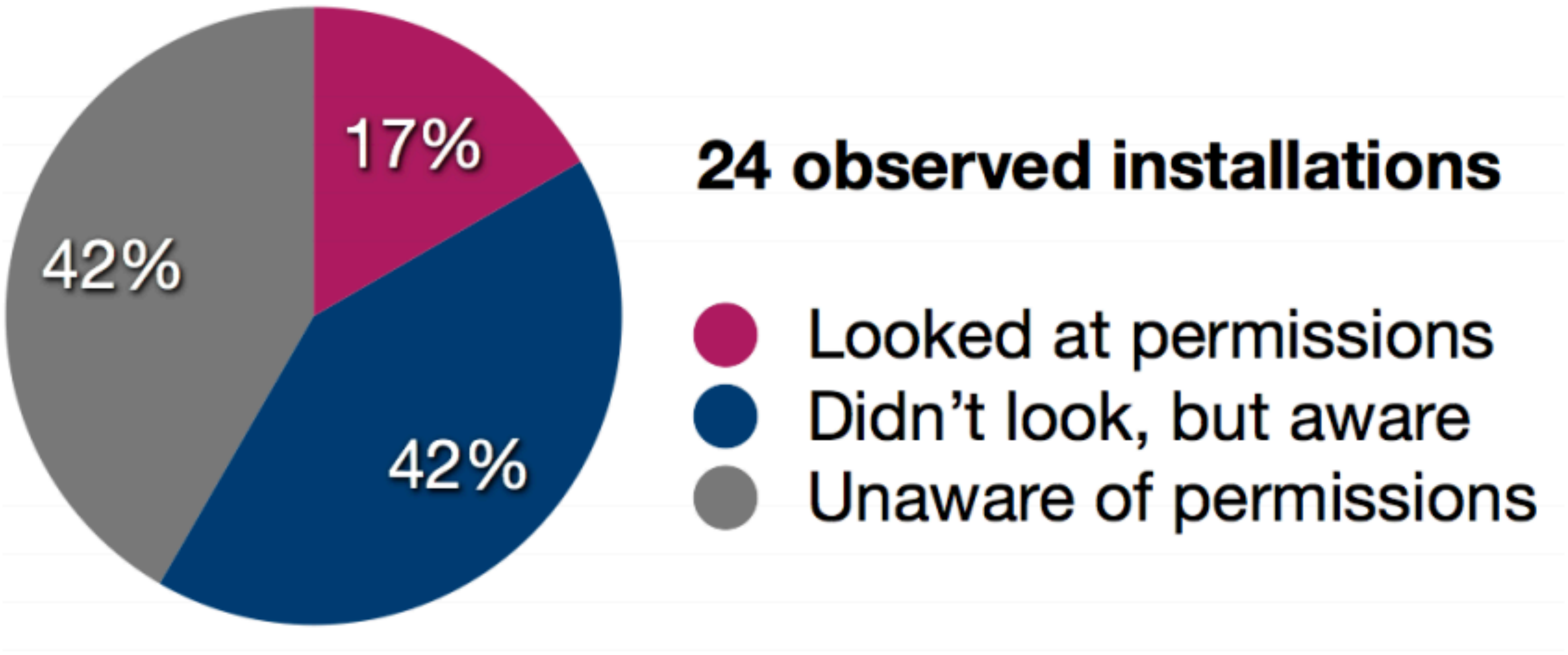
System tools

Network communication

In practice, both are overly permissive:
Once granted permissions, apps can misuse them.

Are Manifests Usable? (Felt et al)

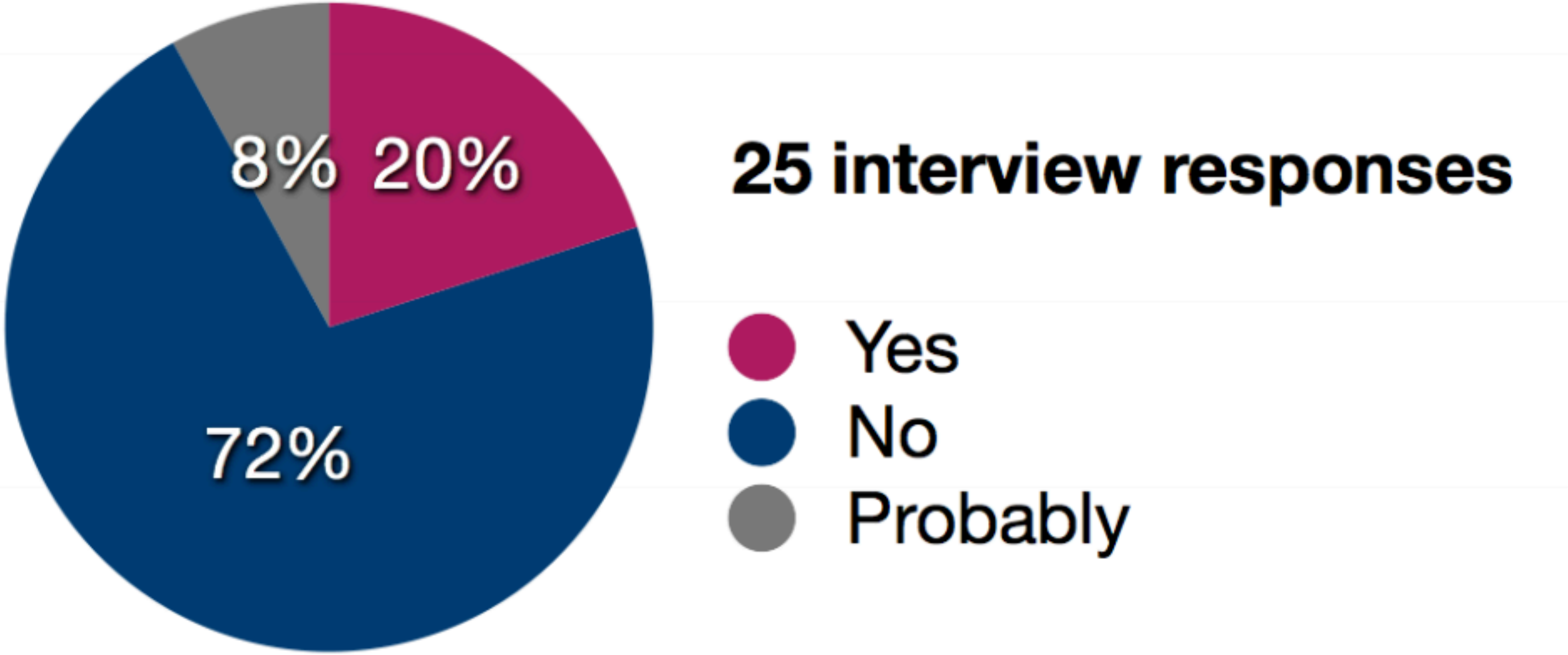
Do users pay attention to permissions?



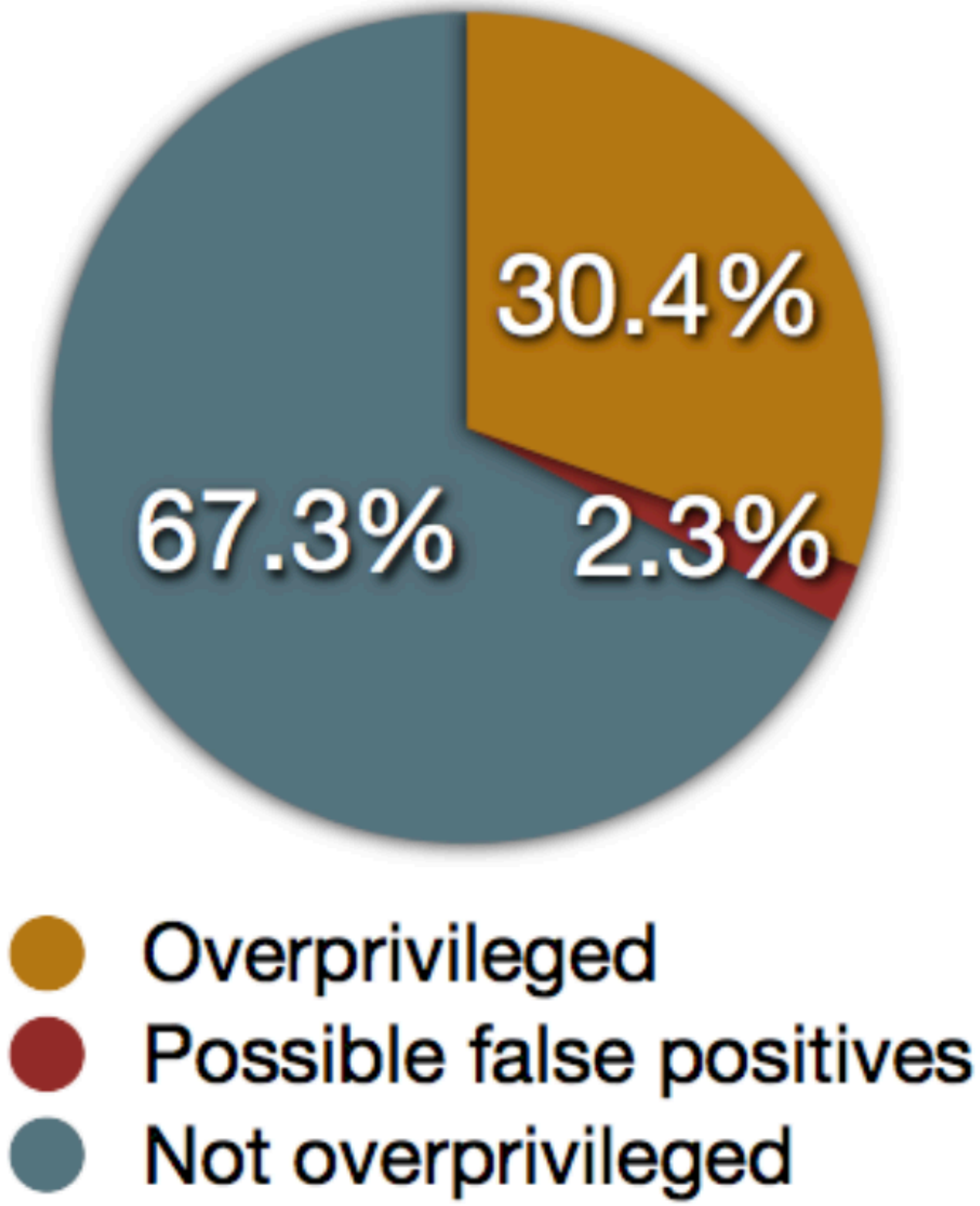
… but 88% of users looked at reviews.

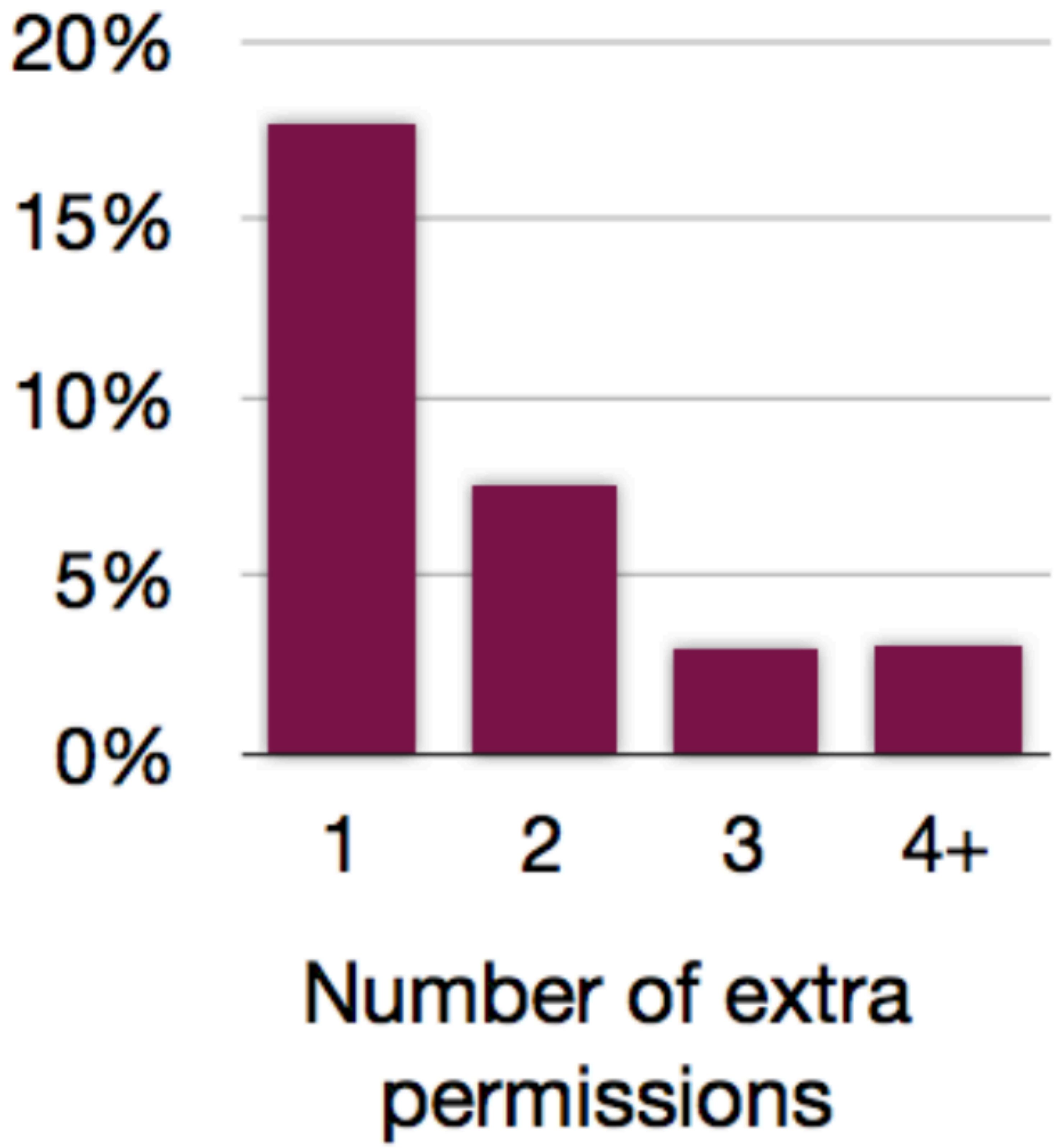
Do users act on permission information?

"Have you ever not installed an app because of permissions?"

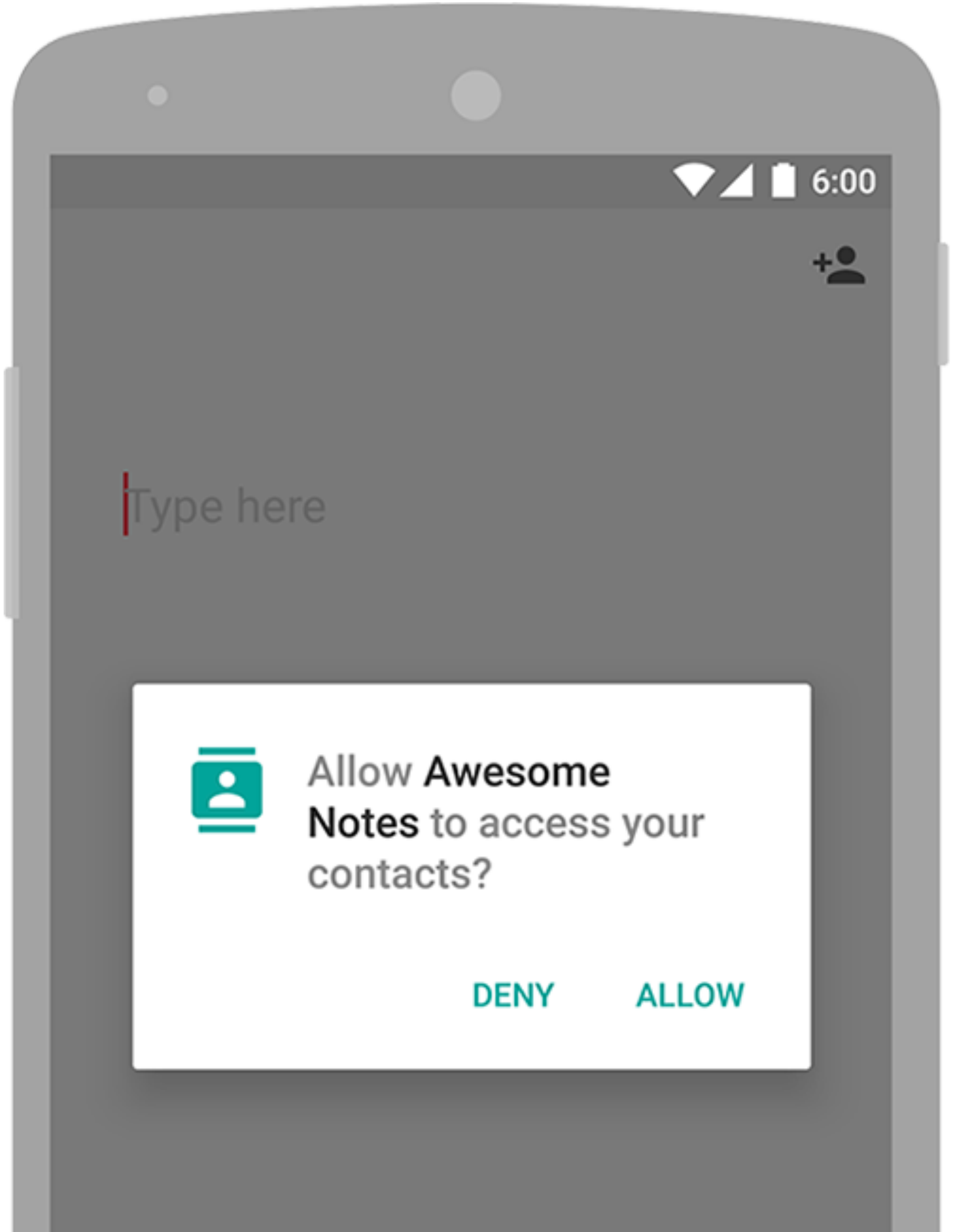


Developers Don’t know the Permissions They Need

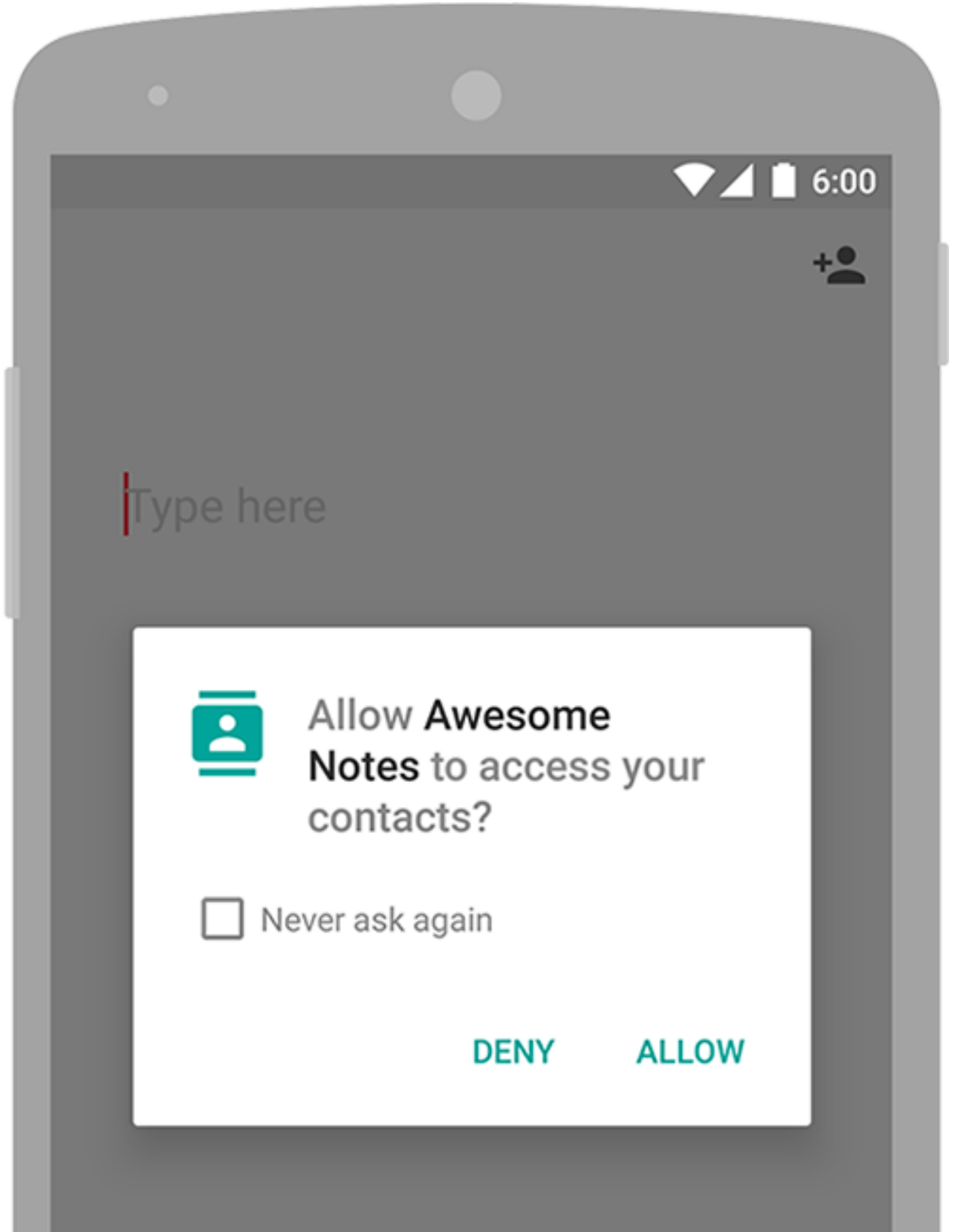




Android Now Asks at Runtime (was not the case historically)



Allow **Awesome Notes** to access your contacts?



Allow **Awesome Notes** to access your contacts?

□ Never ask again

Manifests

In both cases, the Android app needs to request permission in its manifest—it's just up to the Android OS when it asks the user.

The OS might also just grant the right if it doesn't seem dangerous

Manifest also defines what exported endpoints *other* apps can access. Whole class of malware that takes advantage of this of misconfiguration.

Inter-App Communication

Inter-Process Communication

Primary mechanism for IPC between app components in Android: *Intents*

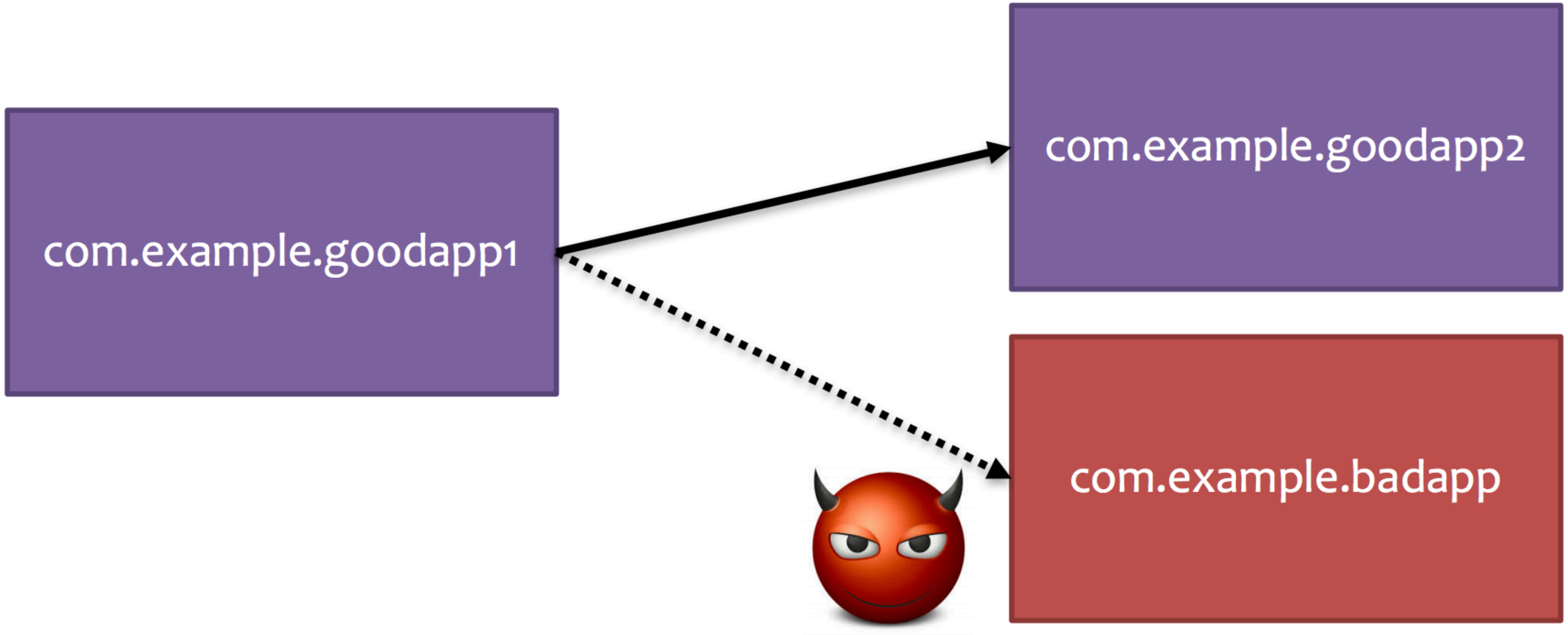
Explicit: specify name: e.g., com.example.testApp.MainActivity

Implicit: Specify action (e.g., ACTION_VIEW) and/or data (URI & MIME type)

An implicit intent specifies an action that can invoke any app on the device able to perform the action. Using an implicit intent is useful when your app cannot perform the action, but other apps probably can and you'd like the user to pick which app to use.

Intent Eavesdropping

Attack #1: Eavesdropping / Broadcast Theft



Unauthorized Intent Receipt

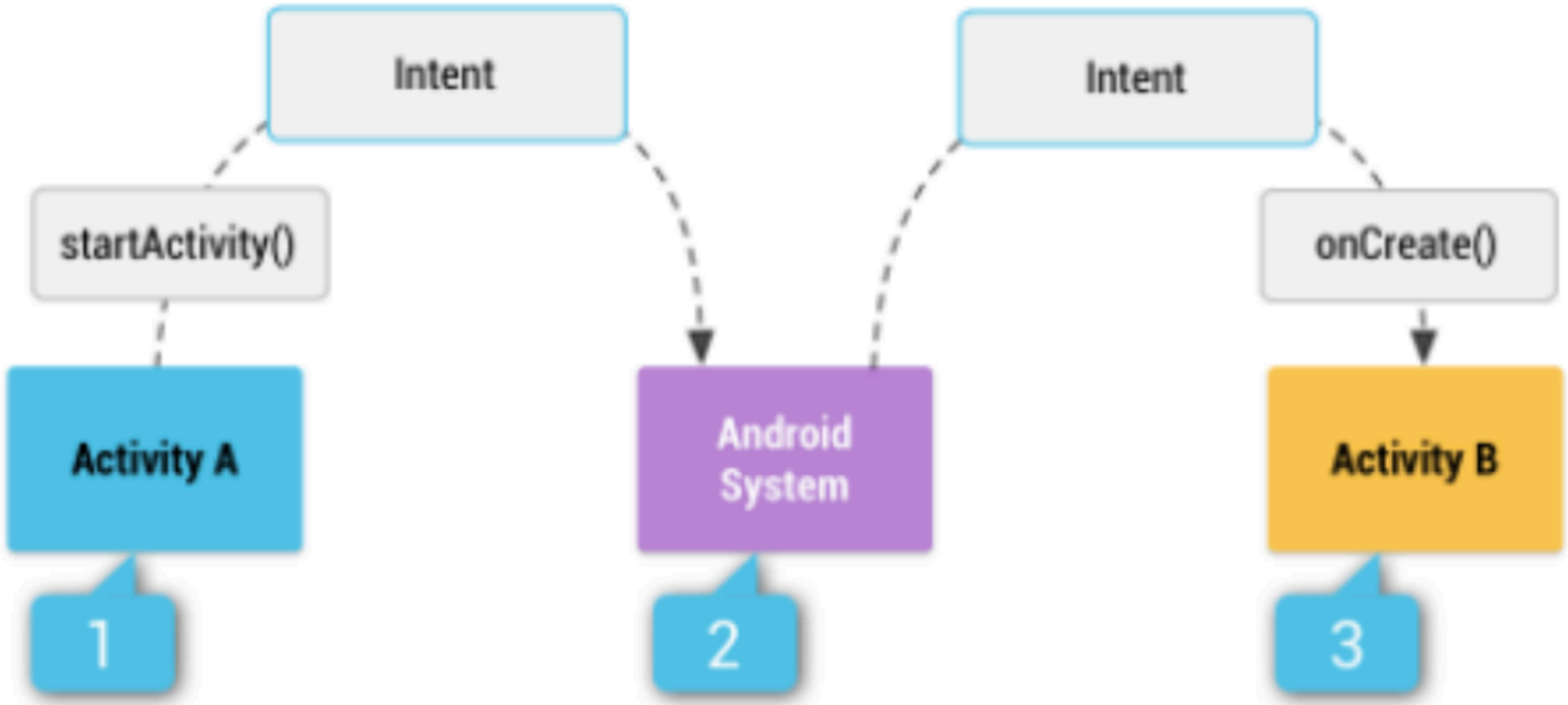
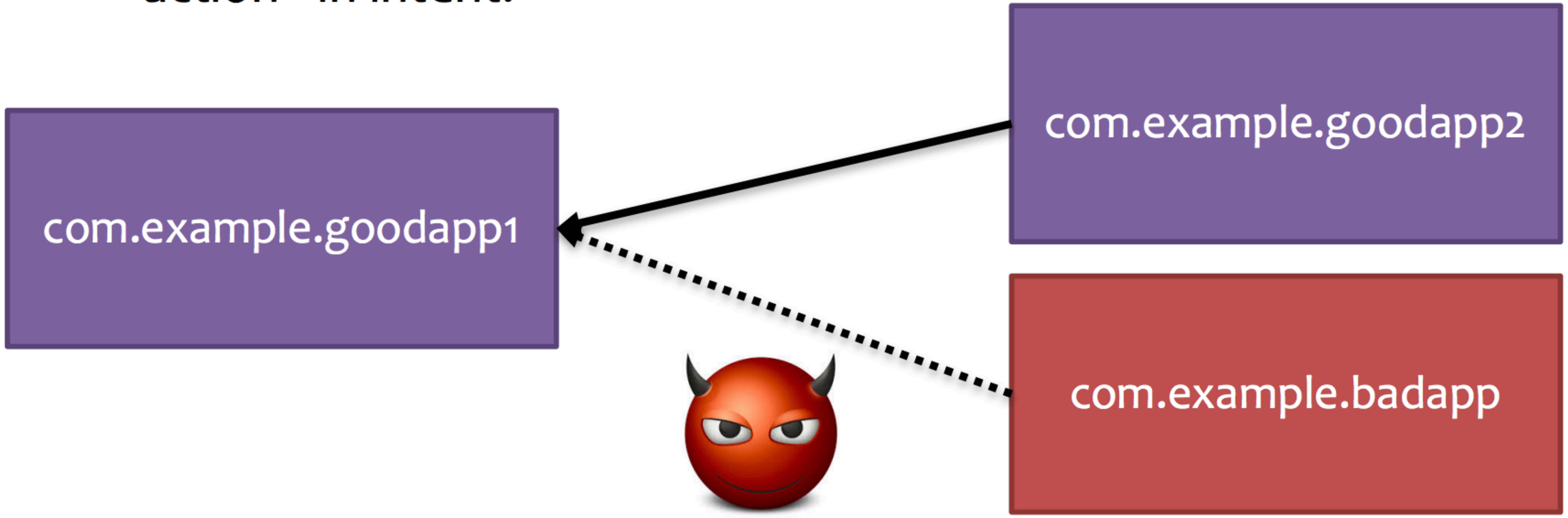


Figure 1. How an implicit intent is delivered through the system to start another activity: **[1]** *Activity A* creates an `Intent` with an action description and passes it to `startActivity()`. **[2]** The Android System searches all apps for an intent filter that matches the intent. When a match is found, **[3]** the system starts the matching activity (*Activity B*) by invoking its `onCreate()` method and passing it the `Intent`.

"**Caution:** To ensure that your app is secure, always use an explicit intent when starting a Servier. Using an implicit intent to start a service is a security hazard because you can't be certain what service will respond to the intent, and the user can't see which service starts."

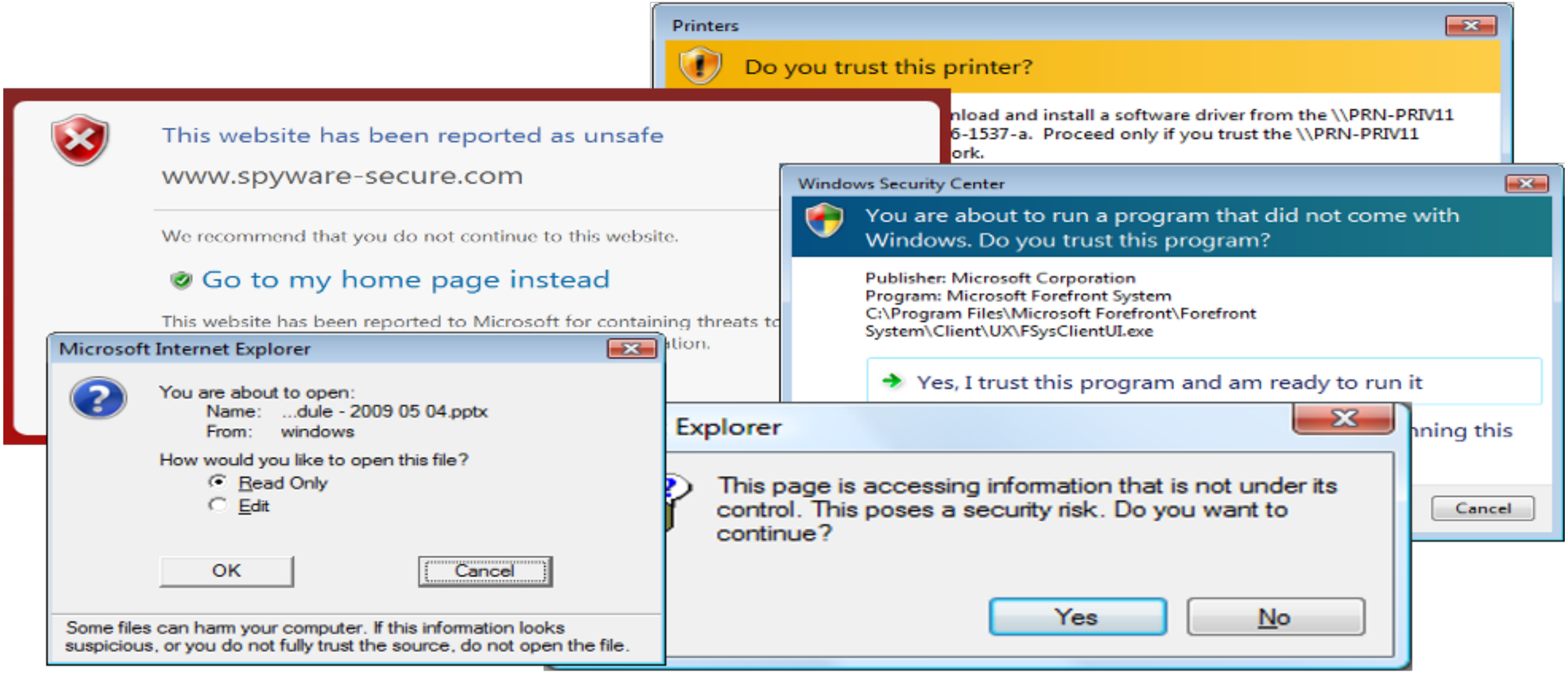
Intent Spoofing

- Attack #1: General intent spoofing
  - Receiving implicit intents makes component public.
  - Allows data injection.
- Attack #2: System intent spoofing
  - Can’t directly spoof, but victim apps often don’t check specific “action” in intent.



Security Dialogues

We inundate users with security alerts



Printers

Do you trust this printer?

nload and install a software driver from the \\PRN-PRIV11
6-1537-a. Proceed only if you trust the \\PRN-PRIV11
ork.

This website has been reported as unsafe

www.spyware-secure.com

We recommend that you do not continue to this website.

Go to my home page instead

This website has been reported to Microsoft for containing threats to
ation.

Windows Security Center

You are about to run a program that did not come with Windows. Do you trust this program?

Publisher: Microsoft Corporation
Program: Microsoft Forefront System
C:\Program Files\Microsoft Forefront\Forefront
System\Client\UX\FSysClientUI.exe

Yes, I trust this program and am ready to run it

nning this

Microsoft Internet Explorer

You are about to open:
Name: ...dule - 2009 05 04.pptx
From: windows

How would you like to open this file?

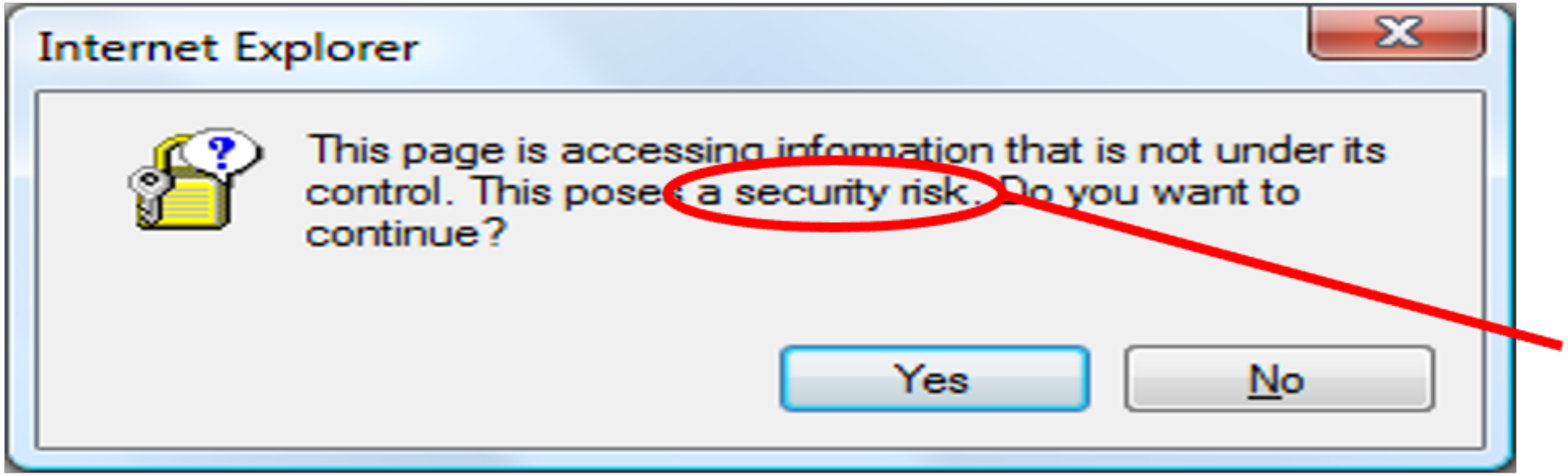
● Read Only
○ Edit

Some files can harm your computer. If this information looks suspicious, or you do not fully trust the source, do not open the file.

Explorer

This page is accessing information that is not under its control. This poses a security risk. Do you want to continue?

Example: IE 6 Mixed Context



Internet Explorer

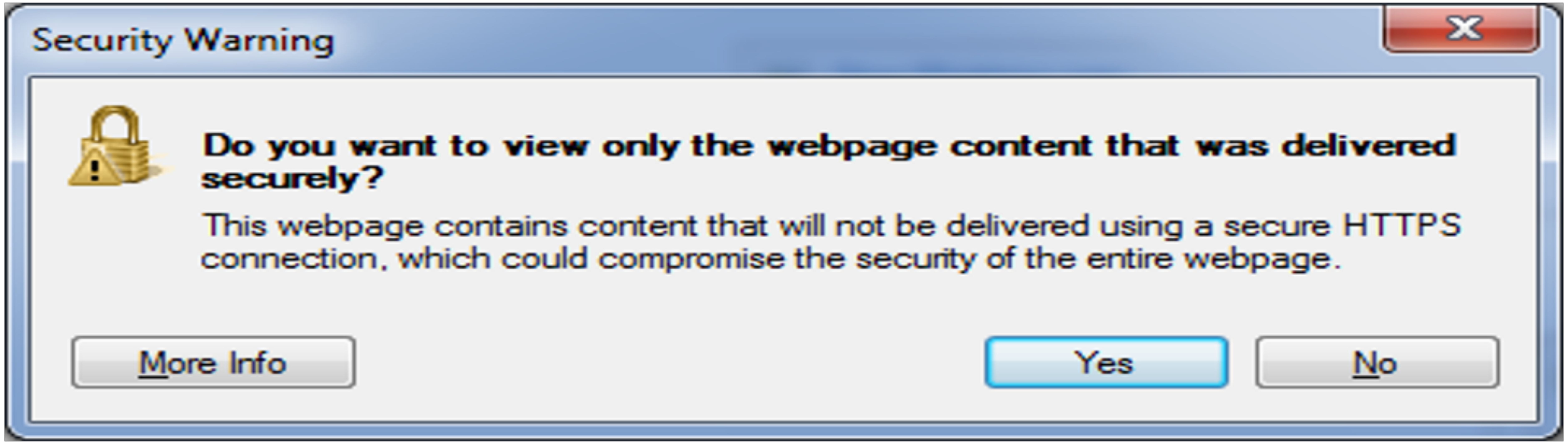
This page is accessing information that is not under its control. This poses a security risk. Do you want to continue?

Vague threat. What's the risk? What could happen?

"Yes", the possibly less safe option, is the default

How should the user make this decision? No clear steps for user to follow.

Example: IE 8 and 9 Improvement



Security Warning

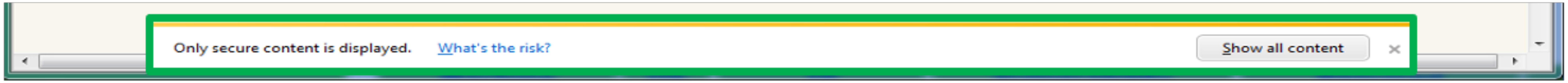
Do you want to view only the webpage content that was delivered securely?

This webpage contains content that will not be delivered using a secure HTTPS connection, which could compromise the security of the entire webpage.

(IE8)

Even better: load the safe content, and use the address bar to enable the rest

(IE9)



Only secure content is displayed.

Interaction Guidelines

Philosophy:

– Does the user have unique knowledge the system doesn’t?
– Don’t involve user if you don’t need to — leads to alert fatigue
– If you involve the user, enable them to make the right decision

Make sure your security dialogs are NEAT:

– *Necessary*: Can the system take action w/o the user? Does user have more information?
– *Explained*: Does the dialogue include enough information to be understandable?
– *Actionable*: Can user make correct decision in malicious and benign situation?
– *Tested*: Test with users who haven’t used system in both malicious/benign situations.

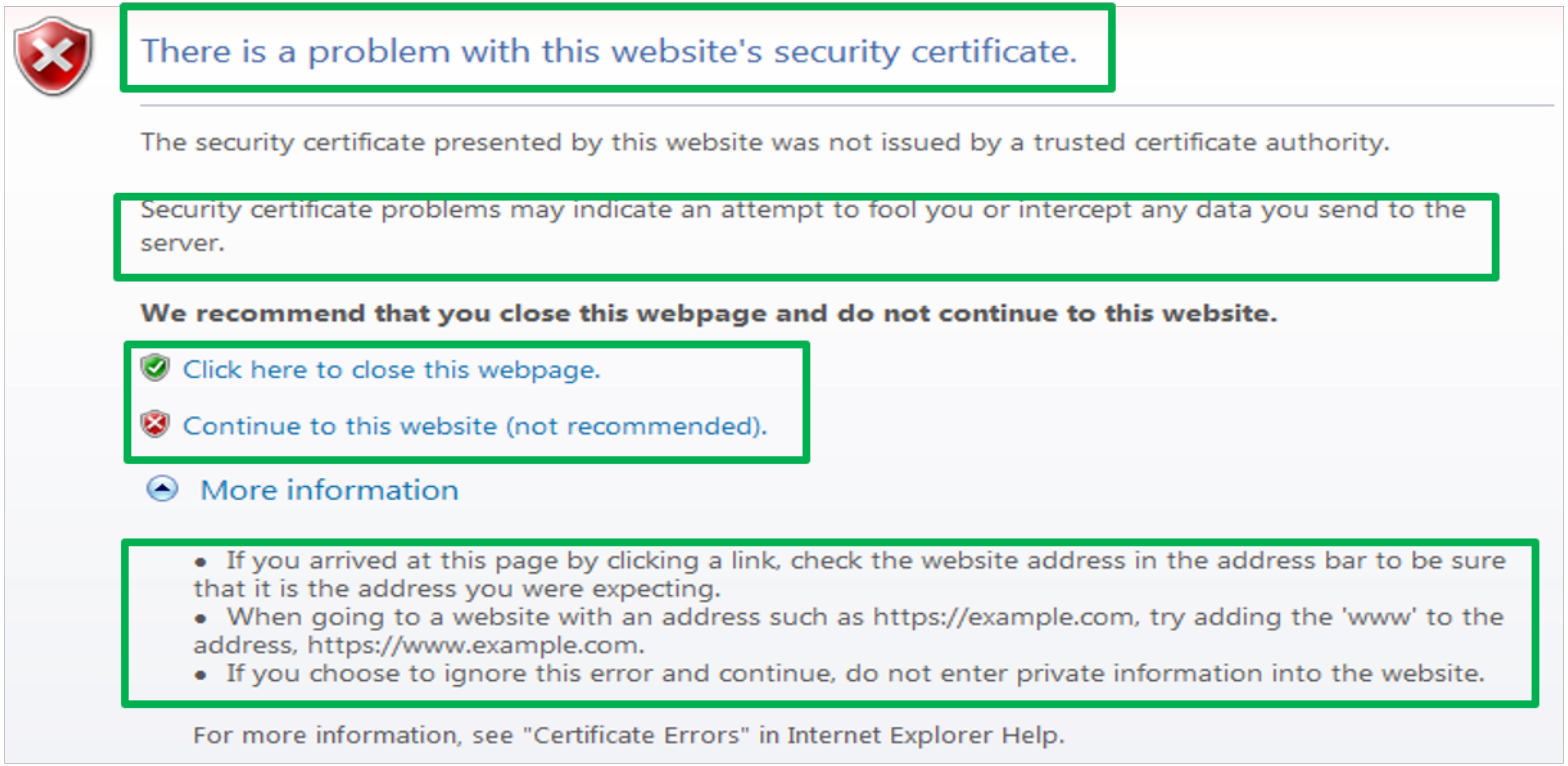
Internet Explorer Bad Certificate Example

Source

Risk

Choices

Process



There is a problem with this website's security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

Click here to close this webpage.

Continue to this website (not recommended).

More information

- If you arrived at this page by clicking a link, check the website address in the address bar to be sure that it is the address you were expecting.
- When going to a website with an address such as https://example.com, try adding the 'www' to the address, https://www.example.com.
- If you choose to ignore this error and continue, do not enter private information into the website.

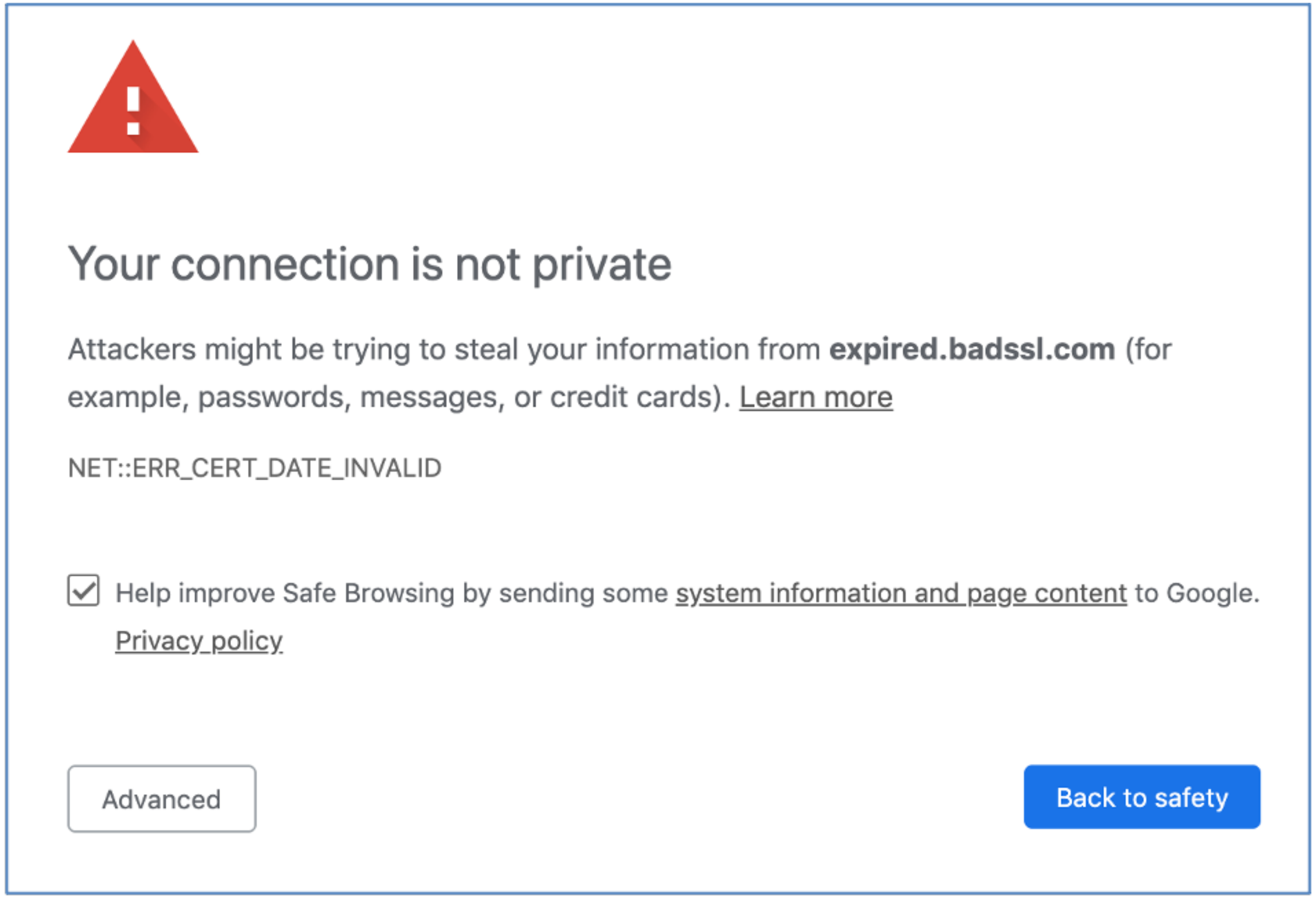
For more information, see "Certificate Errors" in Internet Explorer Help.

Chrome 2019 Bad Certificate Example

Risk

Explanation

Choices



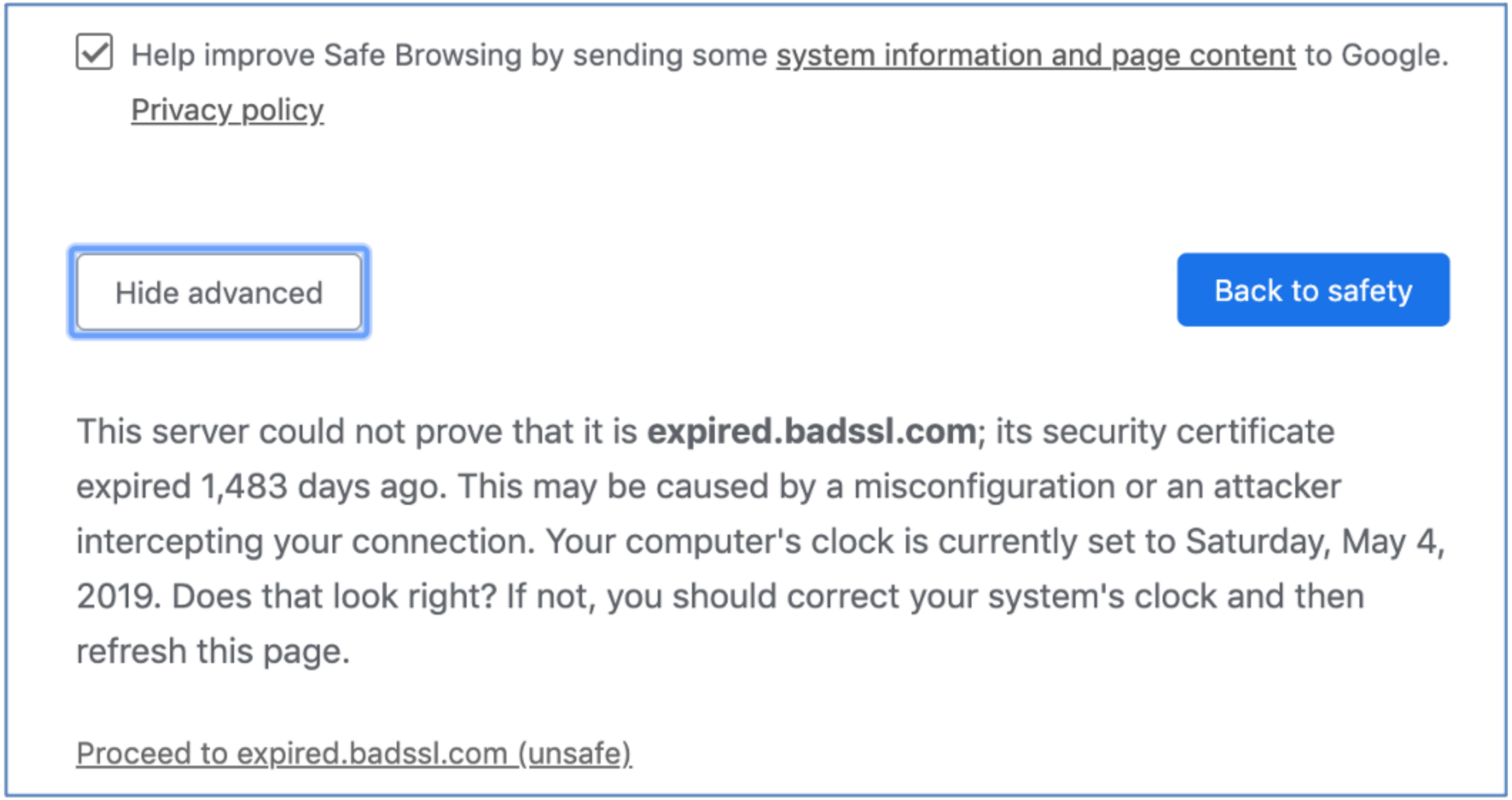
Your connection is not private

Attackers might be trying to steal your information from **expired.badssl.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_DATE_INVALID

☑ Help improve Safe Browsing by sending some system information and page content to Google. Privacy policy

Chrome 2019 Bad Certificate Example



☑ Help improve Safe Browsing by sending some system information and page content to Google. Privacy policy

This server could not prove that it is **expired.badssl.com**; its security certificate expired 1,483 days ago. This may be caused by a misconfiguration or an attacker intercepting your connection. Your computer's clock is currently set to Saturday, May 4, 2019. Does that look right? If not, you should correct your system's clock and then refresh this page.

Proceed to expired.badssl.com (unsafe)

Process

Choice

(expired certificate)

Secure Messaging

Email Protection

Your email provider may be required to turn over your (securely stored) email

- Warrant (for content)

Metadata

- National Security Letter (NSL), Court Order

What if you want to protect email content?

PGP

Modern implementations: GnuPG, Keybase

Each user has:

- A public encryption key, paired with a private decryption key
- A private signature key, paired with a public verification key

How does sending/receiving work?
How do you find out someone's public key?

PGP Operations

To send a message:

Sign with your signature key

Encrypt message and signature with recipient's public encryption key

To receive a message:

Decrypt with your private key to get message and signature

Use sender's public verification key to check sig

PGP Public Keys

How do you obtain Bob's public key?

- Get it from Bob's website? (😖)
- Get it from Bob's website, verify using out-of-band communication
  - Keys are unwieldy fingerprints
  - A fingerprint is a cryptographic hash of a key

- What if you don't personally know Bob?
  - Web of Trust (WoT)
  - Social Network (Keybase)

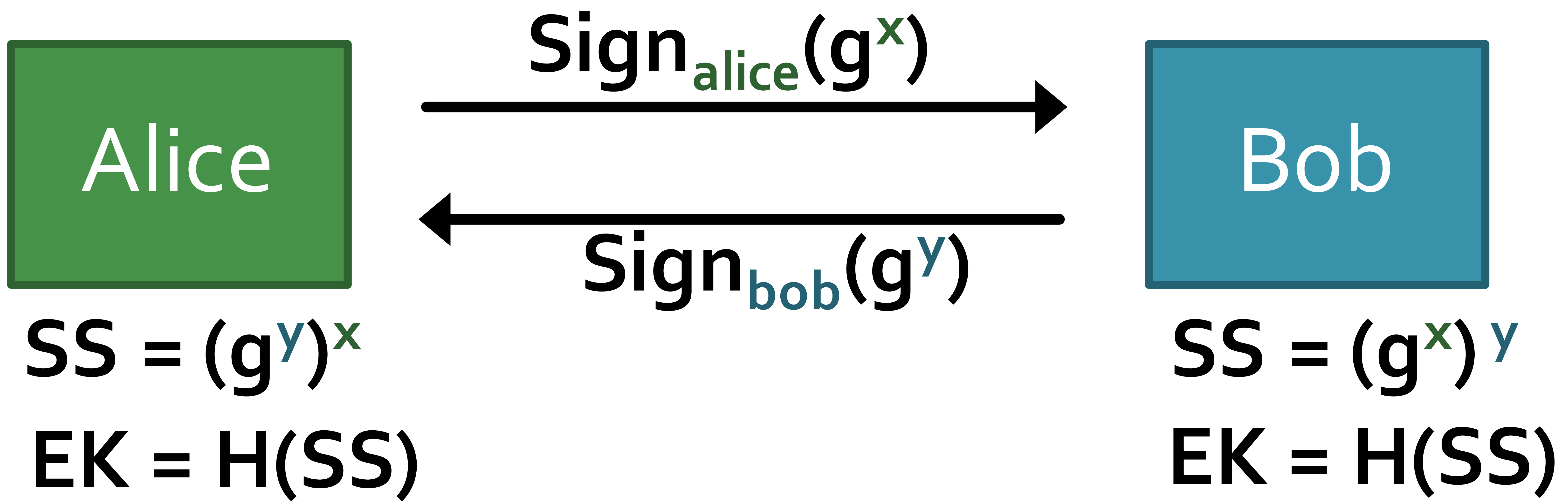
Lost PGP Key

What if Bob's machine compromised?

- His key material becomes known
- Past messages can be decrypted and read
- You also have sender's signature on messages sent, so you can prove identity of sender
- Sender must trust recipient's ability/desire to keep her statements private

OTR: Off the Record Chat

1. Use authenticated Diffie-Hellman to establish a (short-lived) session key EK



Alice

Bob

$\text{Sign}_{alice}(g^{x})$

$\text{Sign}_{bob}(g^{y})$

$SS = (g^{y})^{x}$

$EK = H(SS)$

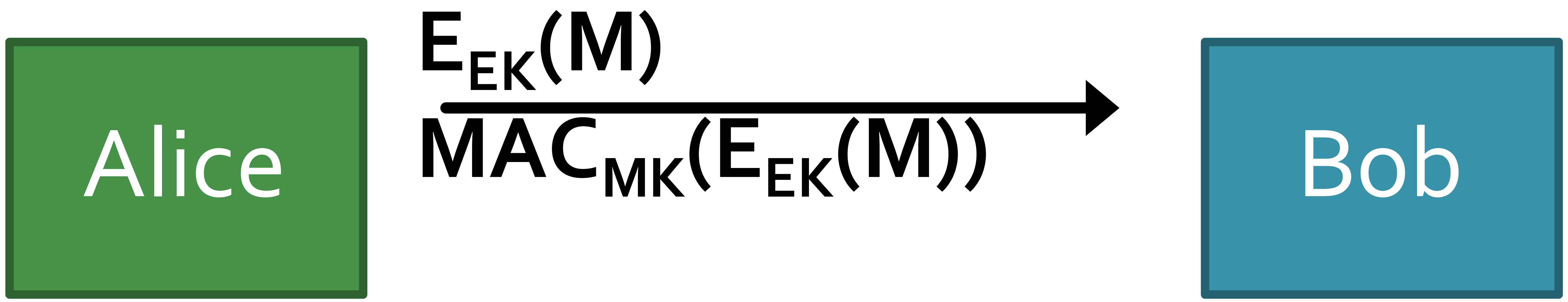
$SS = (g^{x})^{y}$

$EK = H(SS)$

OTR: Off the Record Chat

2. Then use symmetric encryption on message M

... and authenticate using a MAC



Alice

$E_{EK}(M)$

$MAC_{MK}(E_{EK}(M))$

Bob

$SS = (g^y)^x$

$EK = H(SS)$

$\mathbf{MK = H(EK)}$

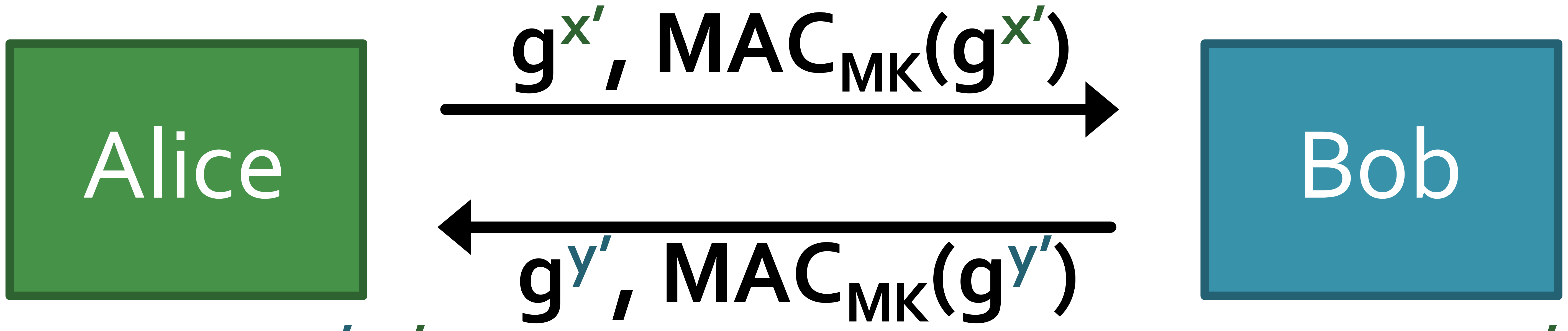
$SS = (g^x)^y$

$EK = H(SS)$

$\mathbf{MK = H(EK)}$

OTR: Off the Record Chat

3. Re-key using Diffie-Hellman



Alice

$g^{x'}, \mathrm{MAC}_{MK}(g^{x'})$

$g^{y'}, \mathrm{MAC}_{MK}(g^{y'})$

Bob

$SS' = (g^{y'})^{x'}$

$EK' = H(SS')$

$MK' = H(EK')$

$MK = H(EK)$

$SS' = (g^{x'})^{y'}$

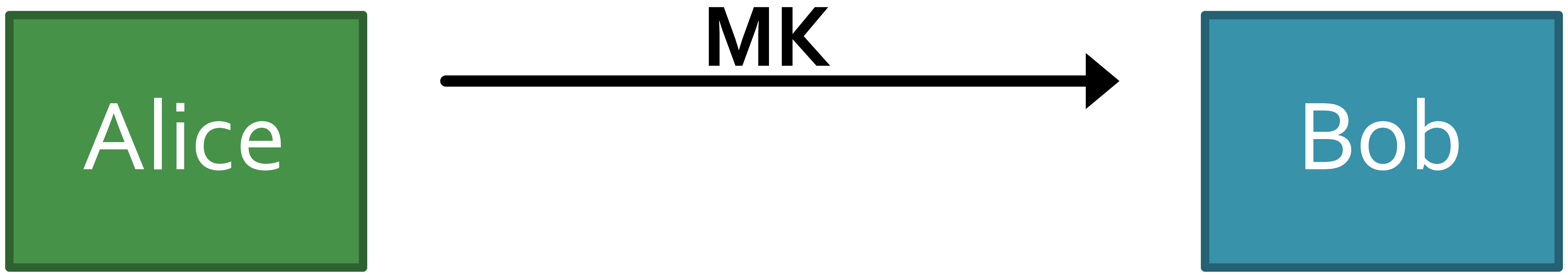
$EK' = H(SS')$

$MK' = H(EK')$

$MK = H(EK)$

OTR: Off the Record Chat

4. Publish old MK



$SS' = (g^{y'})^{x'}$

$EK' = H(SS')$

$MK' = H(EK')$

~~MK = H(EK)~~

"Deniability"

$SS' = (g^{x'})^{y'}$

$EK' = H(SS')$

$MK' = H(EK')$

~~MK = H(EK)~~

Signal/Whatsapp

Note this is suited to interactive communication, not so much email.

But, OTR provides

- message confidentiality
- authentication
- perfect forward secrecy
- deniability

OTR has since lost popularity. Signal Protocol now de facto standard.