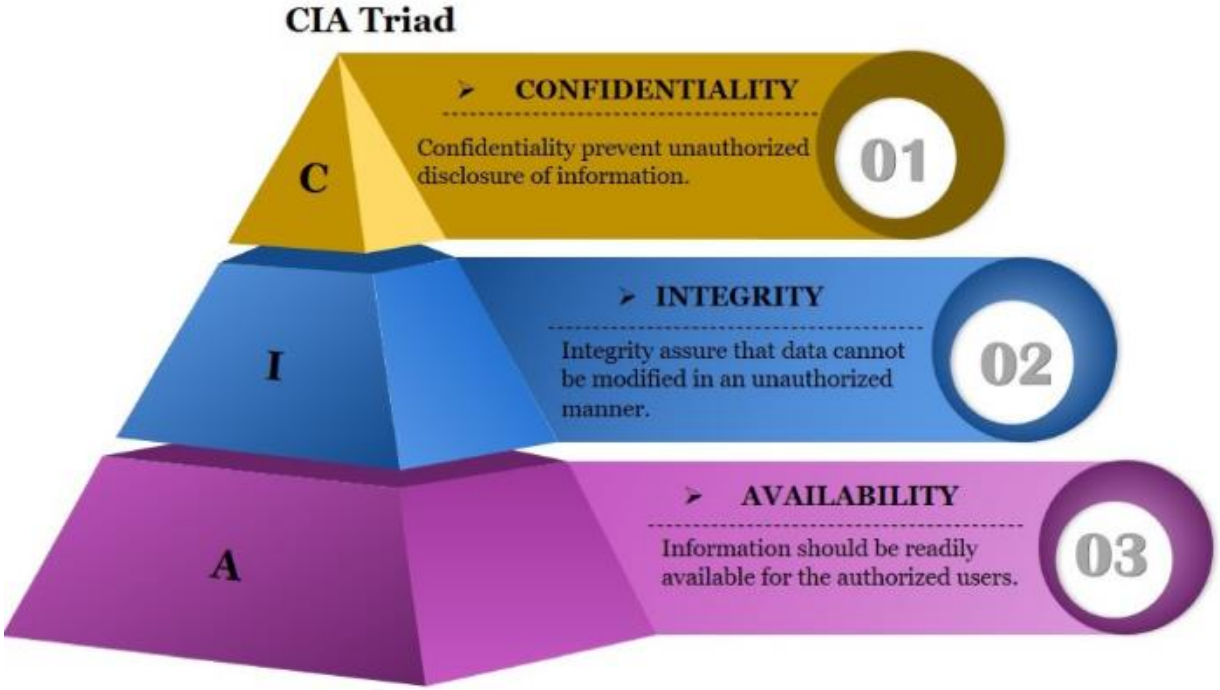
Security for Mobile Devices

CSCI3310

Mobile Computing & Application Development

Security

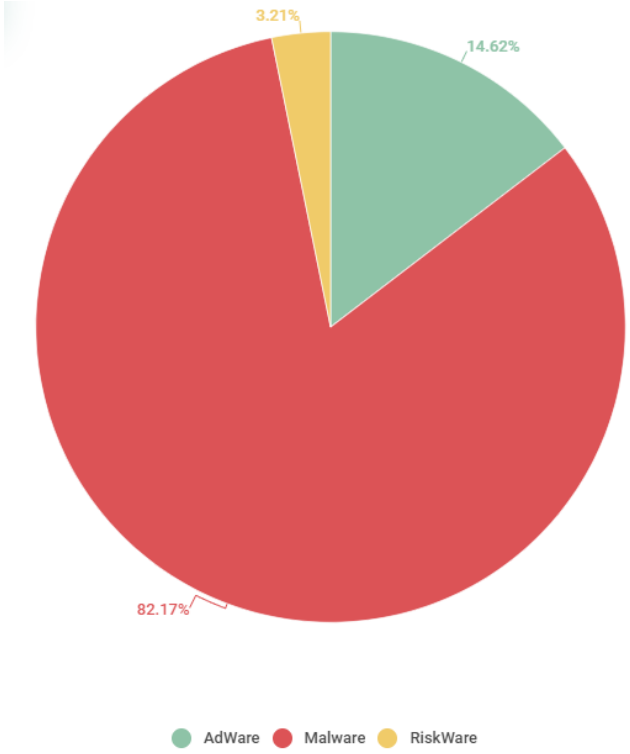
- a state of well-being of information and infra-structures in which the possibility of successful yet undetected theft, tampering, and disruption of information and services is kept low or tolerable
- rests on **C**onfidentiality, **I**ntegrity, and **A**vailability



Mobile Security

- Security on desktop computers is already a well-known issue, smartphone now face the same situation after its widespread usage

Top Mobile Security Threats in 2020
(www.kaspersky.com)



- Malware
  - Virus, trojan, spyware etc. to steal personal, financial information
- App Specific Attack
  - Attack insecurely developed app, e.g., code tampering, injection etc.

OWASP Top 10

The **Open Web Application Security Project** (OWASP)

- an online community, produces freely-available articles, methodologies, documentation, tools, and technologies in the field of *Web Application Security*
- **OWASP Top Ten**, identify top 10 critical risks facing organizations in every few years, published since 2004, aims to raise awareness about application security

OWASP
Open Web Application Security Project

OWASP Mobile Top 10

OWASP Mobile Top 10 is its counterpart in mobile field,

- Latest Top 10 published in 2016
- evolved as the **OWASP Mobile Application Security Verification Standard (MAGVS)**, now version 1.2

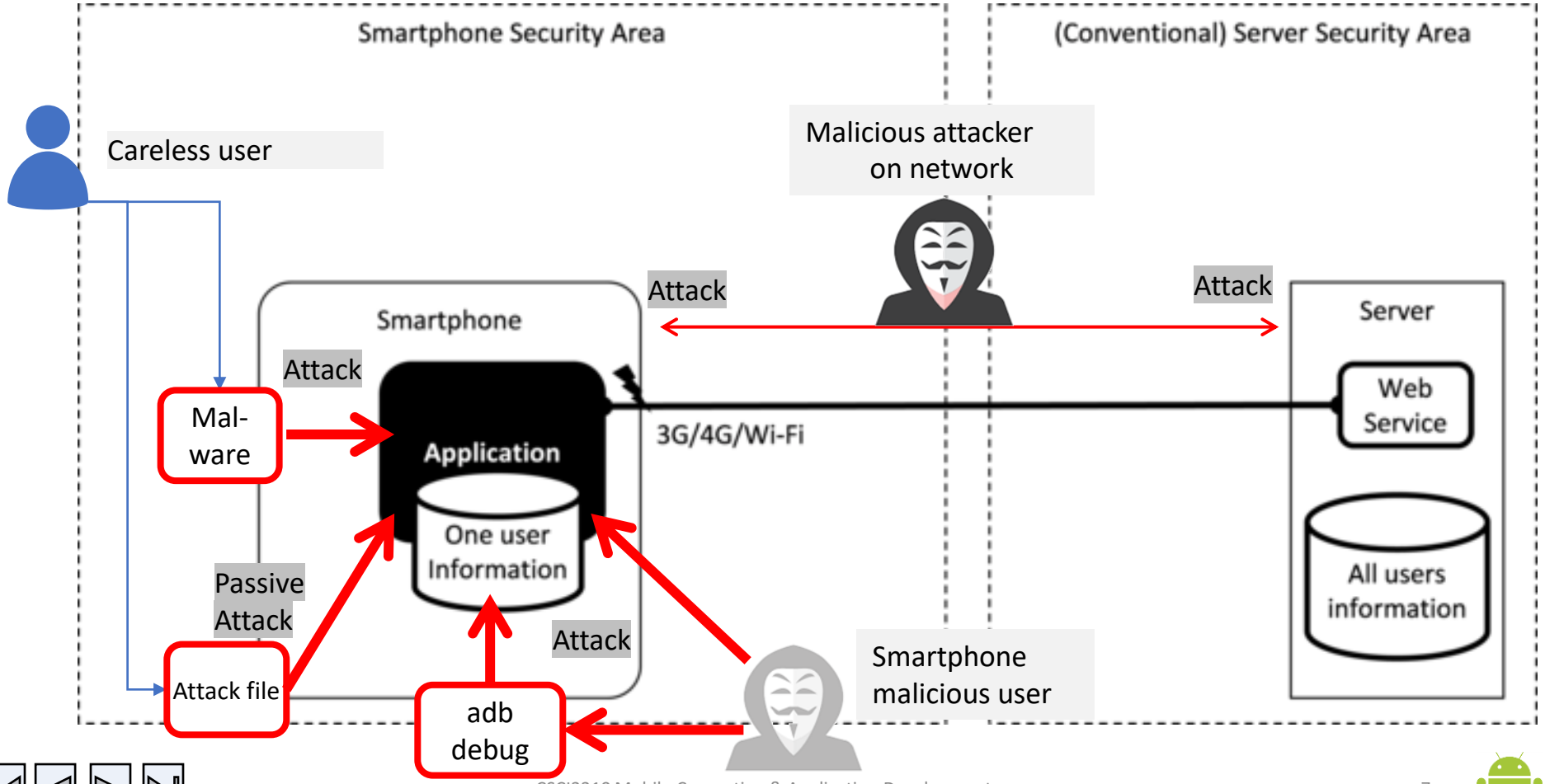
OWASP
Mobile Security Project

OWASP Mobile Top 10

OWASP
Mobile Security Project

Smartphone Security Threats

Various attacks on Smartphone Applications



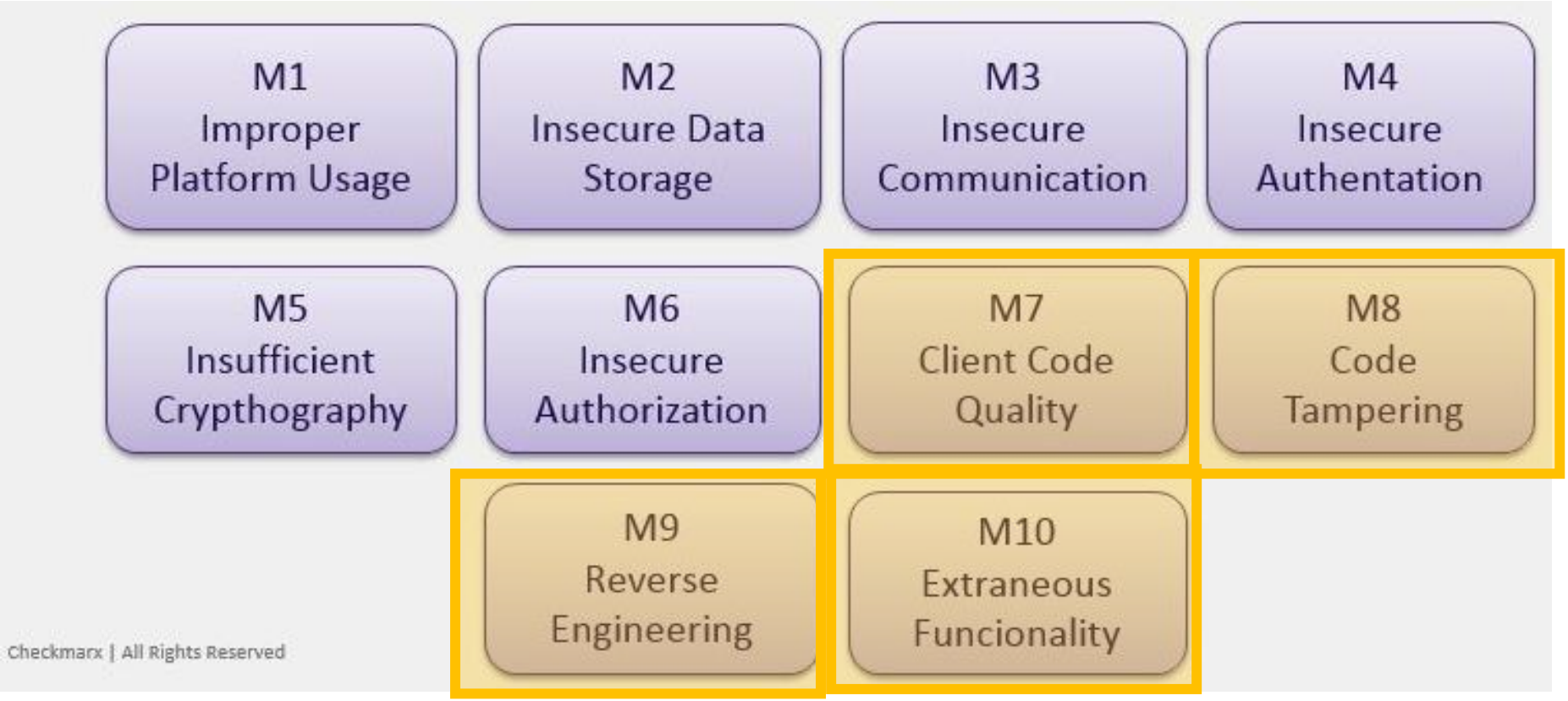
Outline

Common Treats and the combat

- Code Tampering
- Secure Communication
- Data Security
- Platform-related concerns

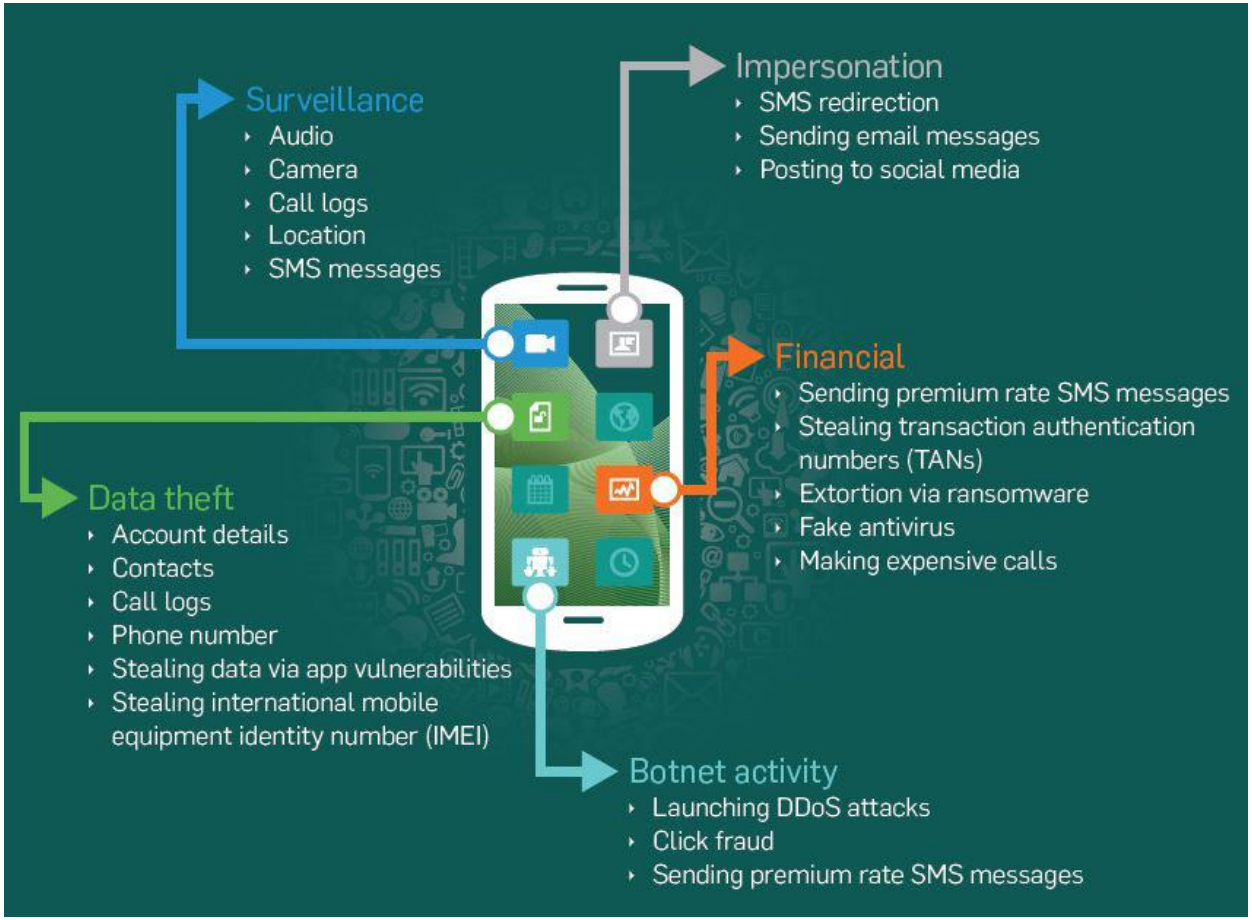
(Optional) Device lost

OWASP Mobile Top 10



OWASP
Mobile Security Project

Malware on Mobile Devices



What mobile malware do

Malware on Mobile Devices

As a developer, malware may infect your app by:

- **Application republishing:** Apps are automatically downloaded, infected with malware, then republished to app stores
- **Malvertising:** Advertisers provide packages of code to allow developers to incorporate ads into their apps.
- **Infected development tools:** a reported case on an infected Xcode can insert malicious functionality into app

How malware finally infected Apple iOS apps: XCodeGhost

Hackers can't easily get malware directly in iOS apps so they're taking a different approach: Modifying the programming environment that Apple provides to make apps.

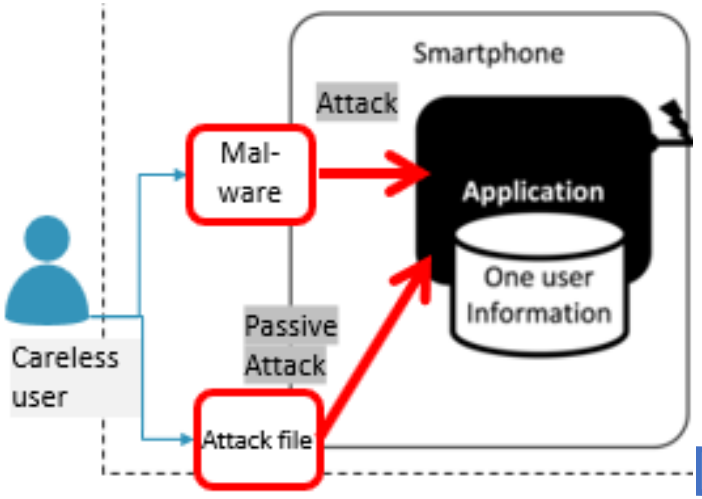
By Kevin Tofel for Mobile Platforms | September 20, 2015 -- 14:41 GMT (22:41 GMT+08:00) | Topic: Apple

Malware or Attack-file Infection methods

1. Drive-by Download
   - Enticing users to download “interesting” media or “feature rich” apps
   - Remote install, require victim’s to input credential and install app into victim’s phone remotely
   - Little did exploiting Android’s bug

2. Auto-run
   - Apps run without being “clicked”
   - Invoked from automated system events
   - Security apps typically scans post install due to framework limitations. This leaves a window open for attackers to exploit

3. Repackaging
   - App phishing
   - Infecting the app
   - Update attach



Android Malware - Repackaging

- Repackaging (App Phishing)
  - Phishing app will then overlay its own interface, tricking the user into entering sensitive information into the phishing app.
  - Rogue apps can masquerade as legitimate apps you trust,
    - e.g., Acecard, a banking trojan faking a "Blackjack" game app. App waits for activity (UI Element) of interest to spawn.

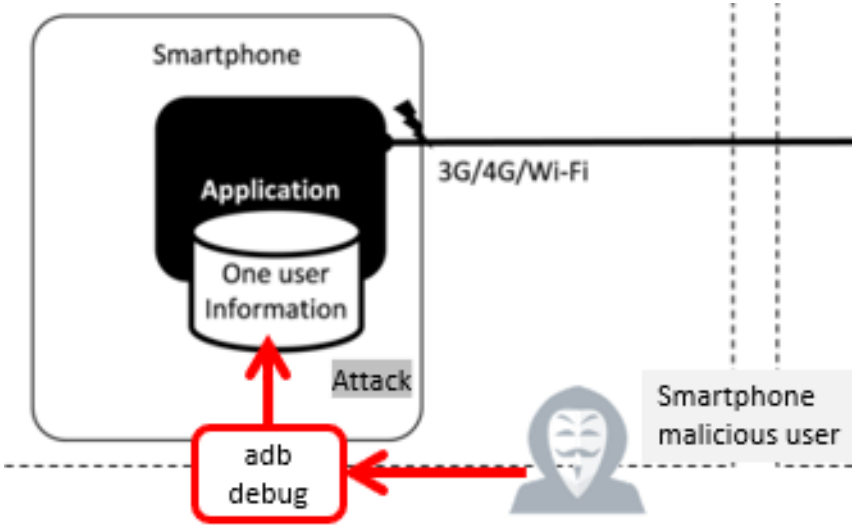
A fake Angry Bird in Space game, hides itself by masquerading as a functional Jpeg file, to gain root Android permissions for carrying out malicious acts.

Android Malware - Repackaging

- Repackaging (Infecting legitimate apps)
  - Android apps are deployed using **APK** file
  - Tools available to disassemble **APK** into its original constituents (classes, manifest, …)
  - By modifying the AndroidManifest.xml, program's entry point can be modified to malware own class
  - After repackaging the modifying/additional files, the malware author can sign the package with their own keys and upload to any outlets e.g., *Google Play* or 3rd party hosting

Android Hacking Tools

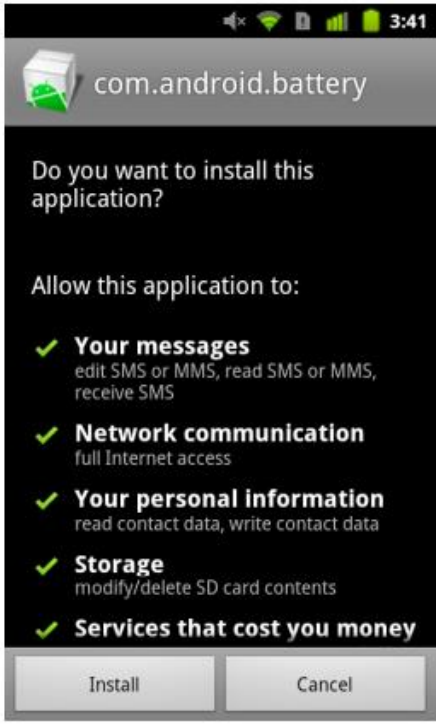
- **DEX2JAR –** Convert compiled DEX object code to a JAR that can be decompiled with JAD.
- **APKTOOL – Disassembler and binary xml** translator built in. Produces Jasmin like syntax that can be reviewed by your favorite editor. Also supports apk rebuilding.
- **JD-GUI** - part of "*Java Decompiler project*" which decompile Java "byte code" and analyze with a standalone graphical utility that displays Java ".class" source codes.



Android Malware - Repacking

Repackaging (Update attack)

- Repackaging piggybacks the mal-code into host, makes it easy to detect
- Includes as an update component that fetch malicious payloads at runtime
- Static scanning of host apps fail to capture the malware



Combating Repackaging

Self-Signing Restriction

- Apply app signing from self-signing to signed-by-market and prevent app distribution without the market's signature.

Code Attestation

- Integrity checking of binary execution code be done by market e.g., SafetyNet Attestation API or a third party

Code Obfuscation

- A technique used for making reverse engineering of source code or machine code more difficult

Combating Reverse Engineering

Java Obfuscation

- By use of obfuscation, it usually slows down reverse engineering significantly

ProGuard in Android

- a Java Optimizer and Obfuscator
- renames classes, fields, and methods using short *meaningless* names
- making code more compact, more efficient, and more difficult to reverse engineer

Combating Reverse Engineering

- Consider writing the **security-sensitive** features, such as *licensing verification* of the code in **C/C++** (i.e., **NDK**) and add them as a compiled library

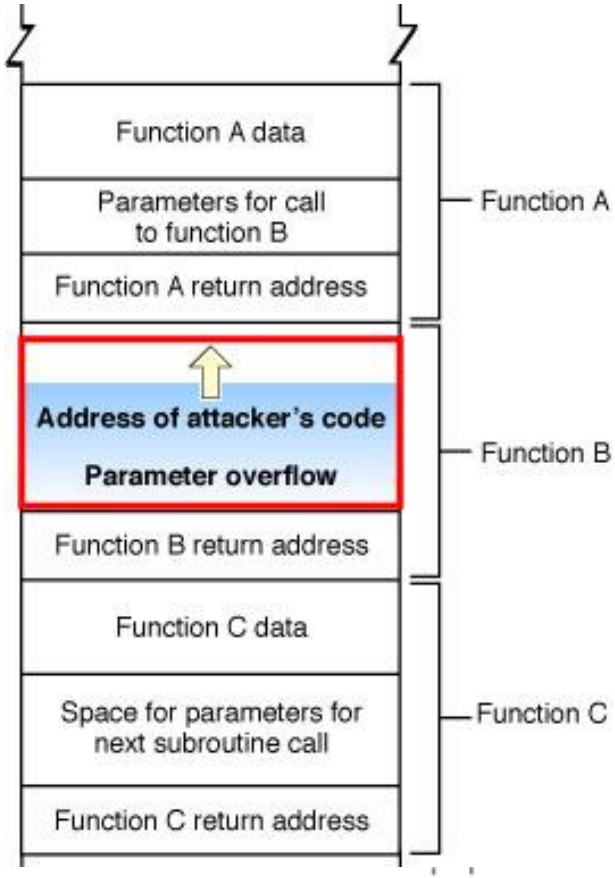
- They can be disassembled into assembly code, but reverse-engineering a large library from assembly is extremely time consuming

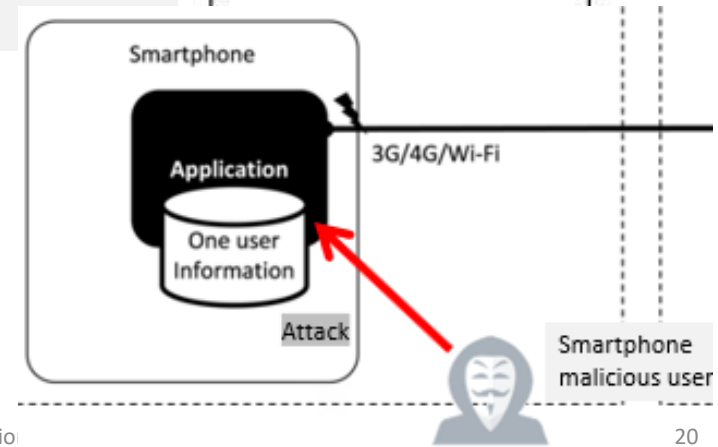
NOTE: be reminded of the ***Buffer Overflows*** vulnerability

Buffer Overflows

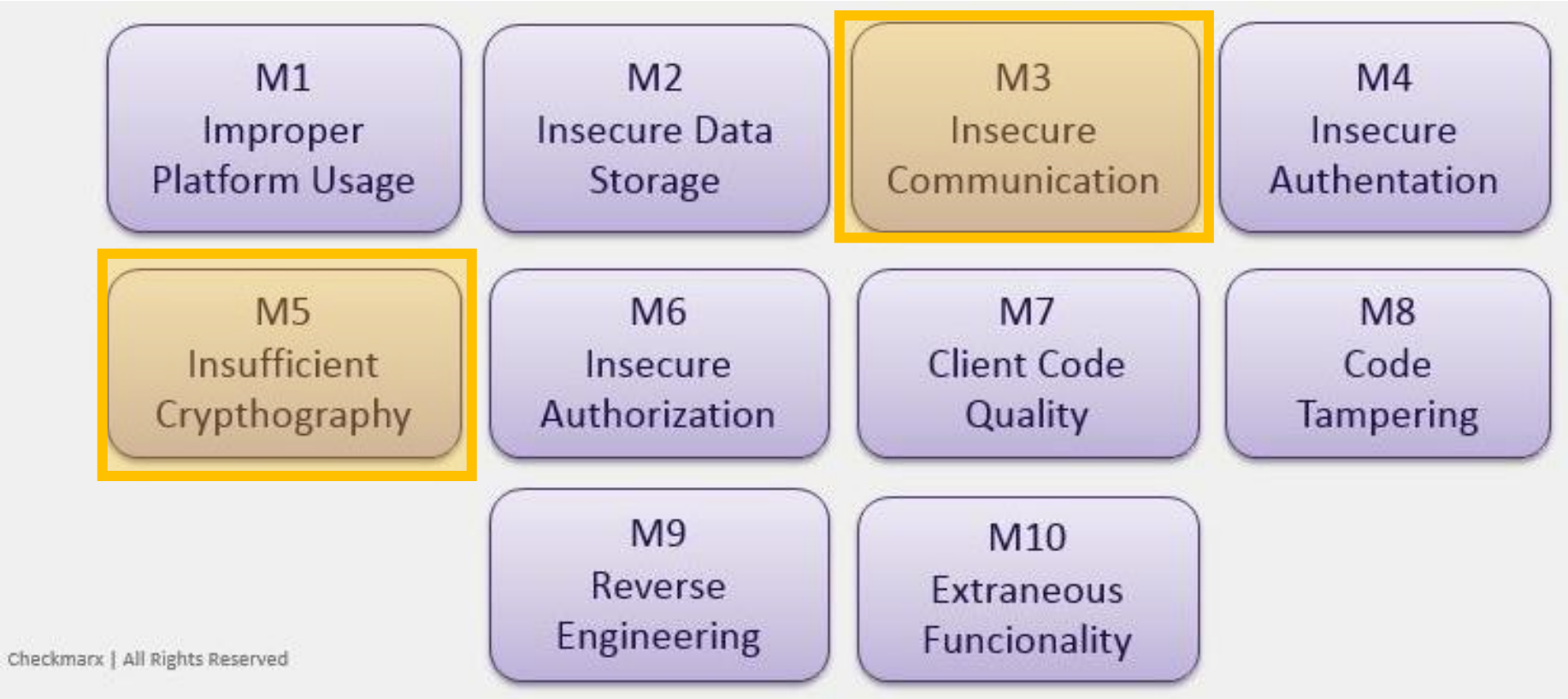
- Buffer overflows, both on the stack and on the heap, are a major source of security vulnerabilities in **C**, **Objective-C**, and **C++** code.

```c
int main(int argc, char **argv)
{
  char buf[8]; // buffer for eight characters
  gets(buf);   // read from stdio (sensitive function!)
  printf("%s\n", buf); // print out data stored in buf
  return 0;    // 0 as return value
}
```





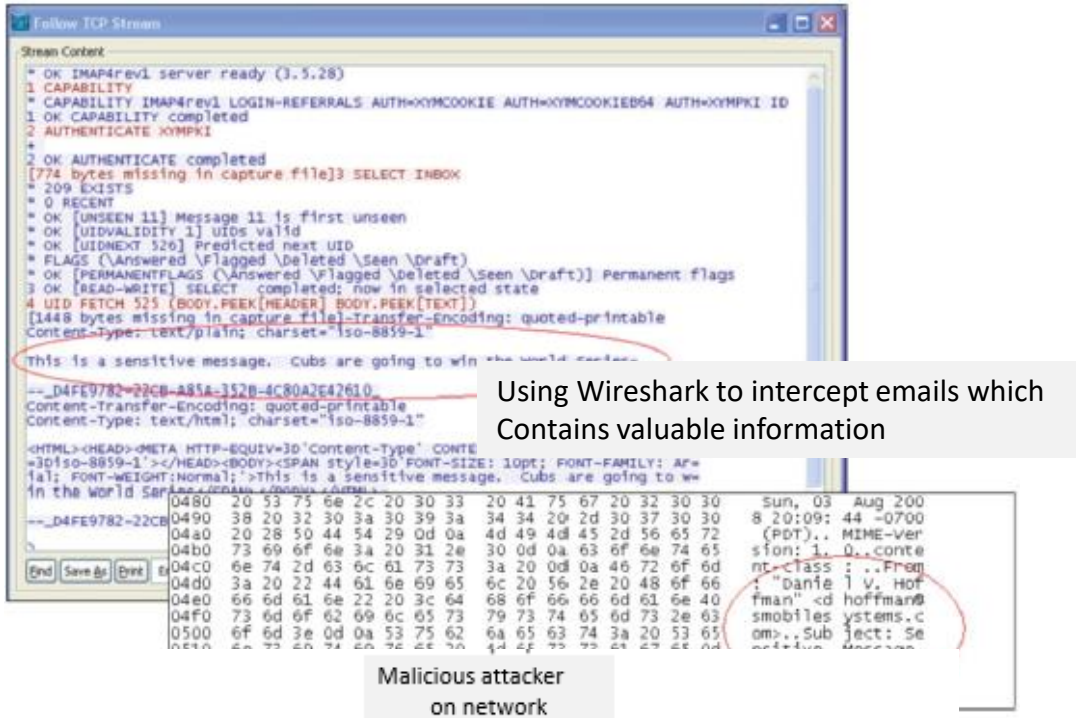
OWASP Mobile Top 10



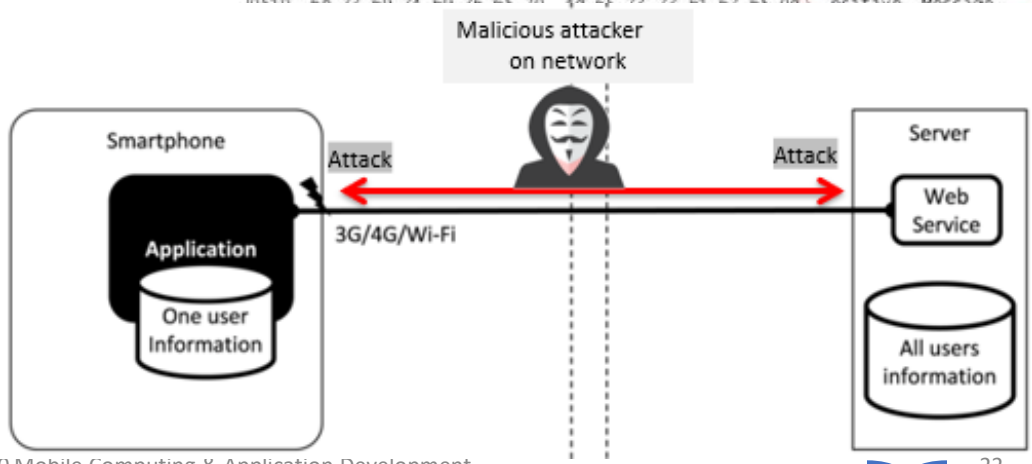
OWASP
Mobile Security Project

Protecting Public Network Traffic

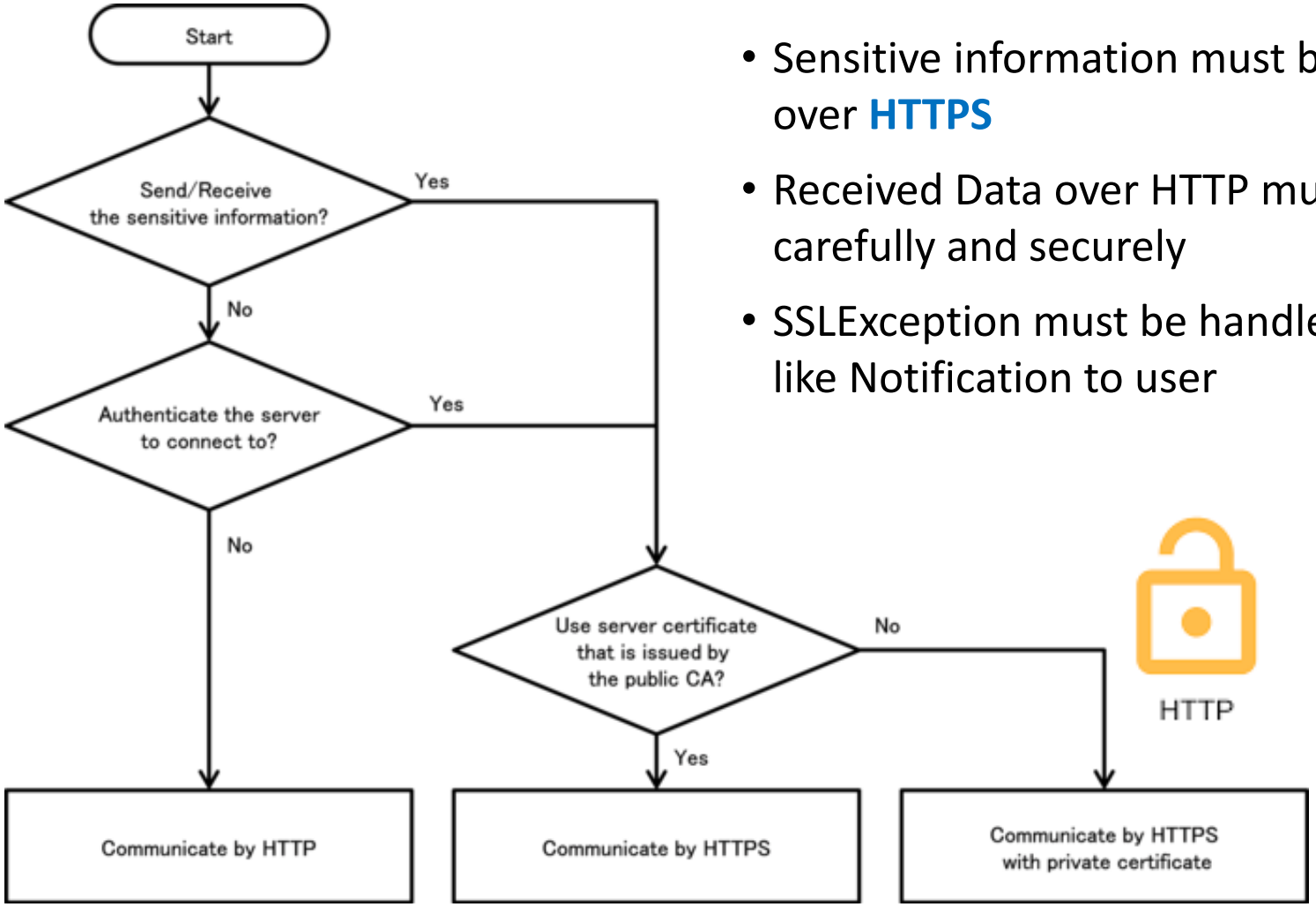
- Man-in-the-Middle Attacks
  - Hackers insert themselves into the communication stream from a device connected to unsecure WiFi network
  - By logging relayed information, criminals can access sensitive user and corporate information



Using Wireshark to intercept emails which Contains valuable information



Communicating via HTTPS

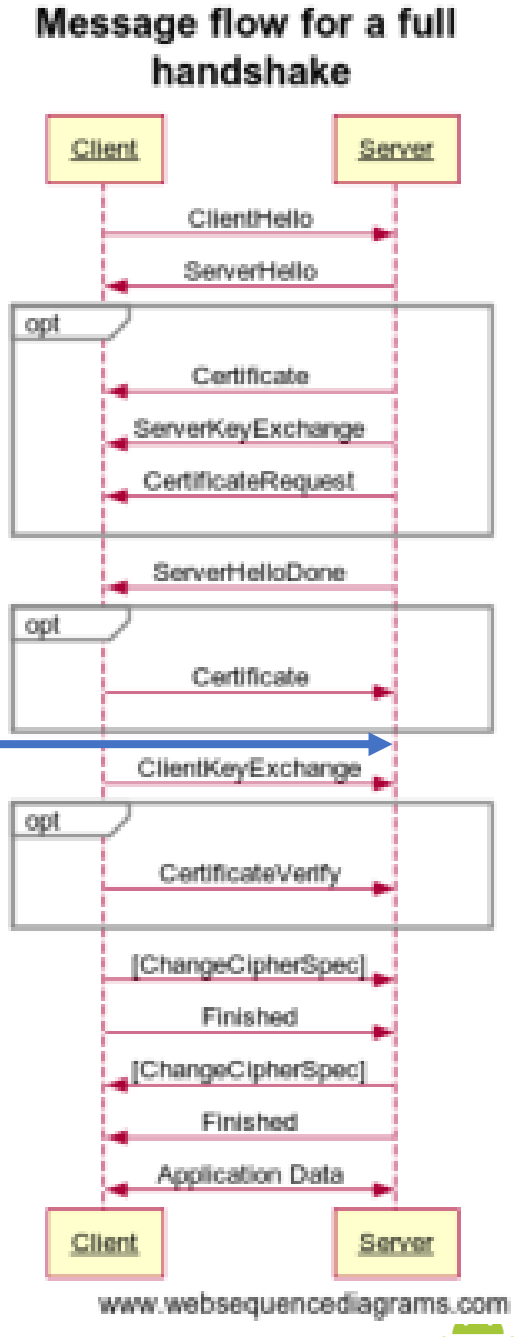


- Sensitive information must be sent/received over **HTTPS**
- Received Data over HTTP must be handled carefully and securely
- SSLException must be handled appropriately like Notification to user

But… how about SSL Exploits?

- A vulnerability was discovered in OpenSSL (CVE-2014-0224)‡ that can leave apps open to a "man-in-the-middle" attack that decrypts secure traffic without either side knowing.

The bug finds its own start if a ChangeCipherSpec message is injected after the ServerHello but before the master secret has been generated

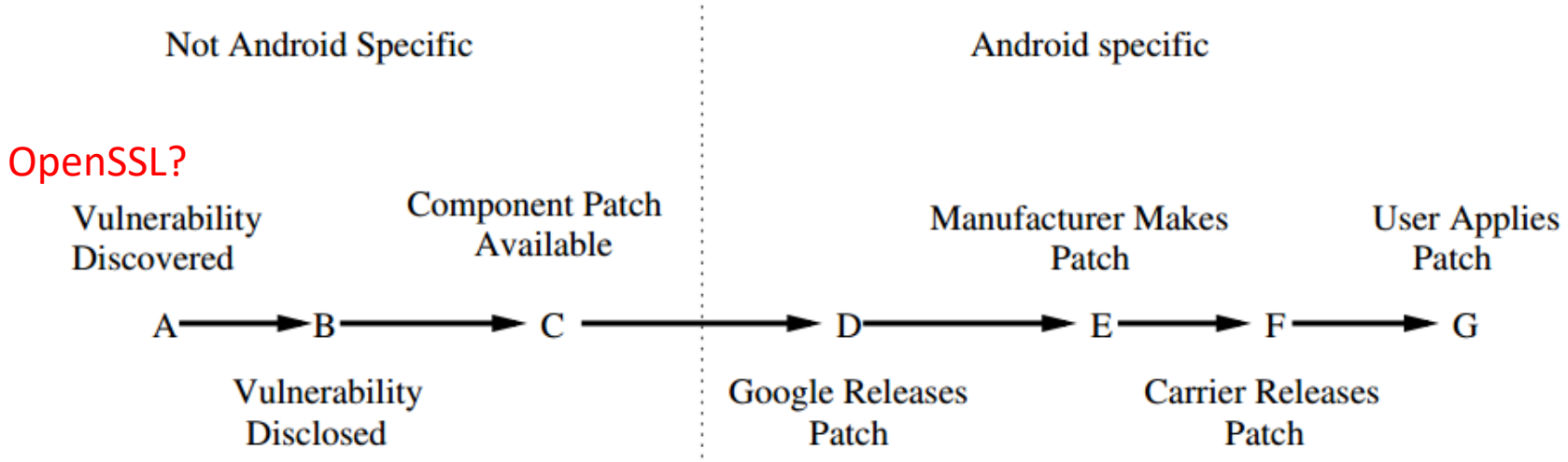


‡ http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0224

https://marcoramilli.com/2014/07/02/openssl-ccs-attack/

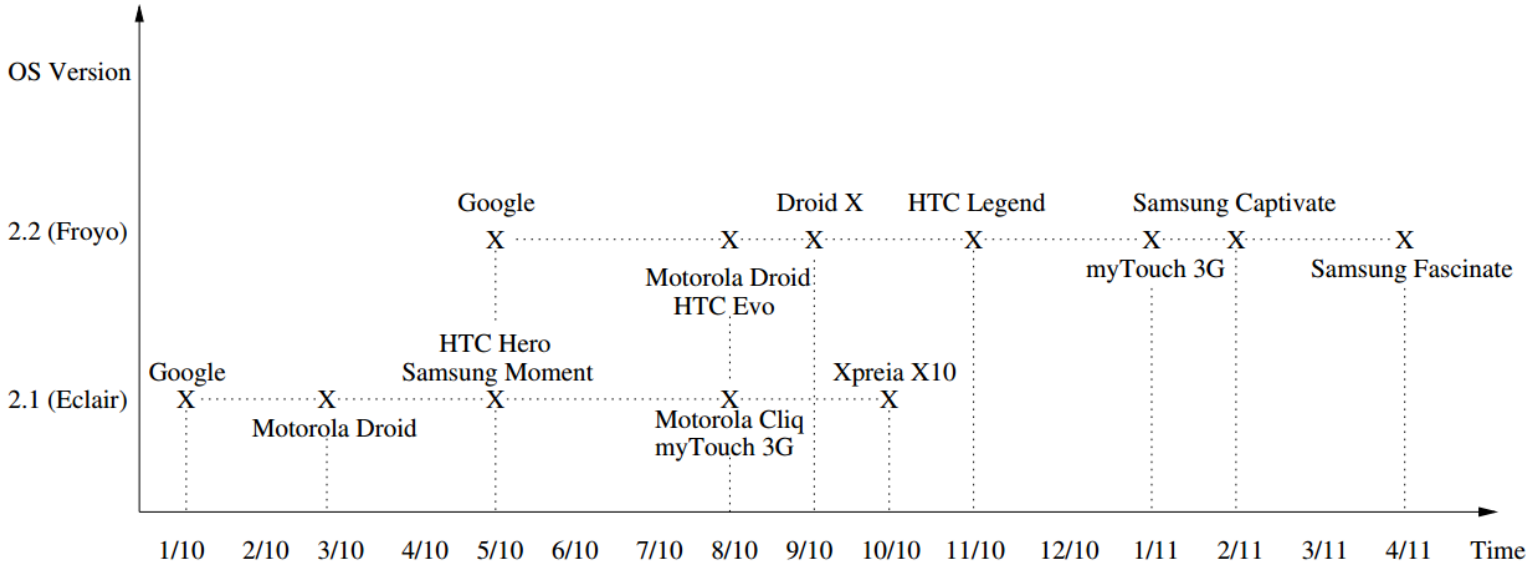
Android Fragmentation

- Android open-source model relies on vendor to push security patches
  - Customized version of Android by vendors need even longer patching time
  - Time window again here



Android Fragmentation

- **Updates** to Android **may never be made available** to the user if the carrier deems deployment too costly
- Even made, a delay still exist before the patch is released by the carriers



Effects of Fragmentation

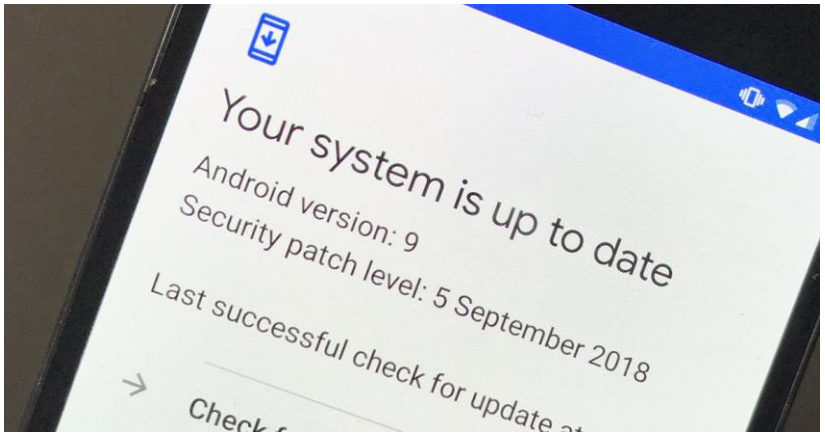
- Many vulnerabilities only present on a single device model or a subset of device models

- Physical devices become a requirement!

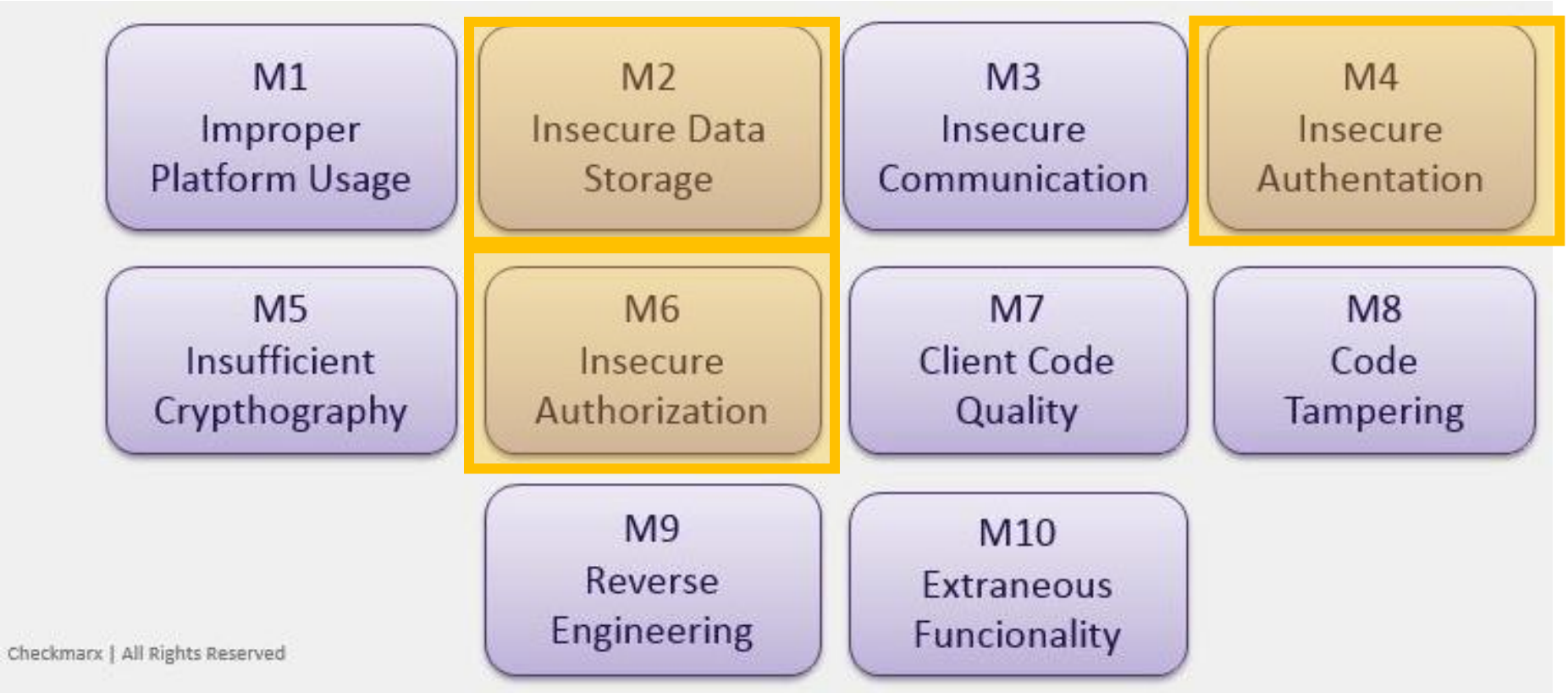
Secure Communication

As a developer

- Protect app from “Man-in-the-Middle” secure traffic decryption
- Use the ProviderInstaller class to catch Exception if the device's Google Play services library is out of date, and patch security provider synchronously or asynchronously.



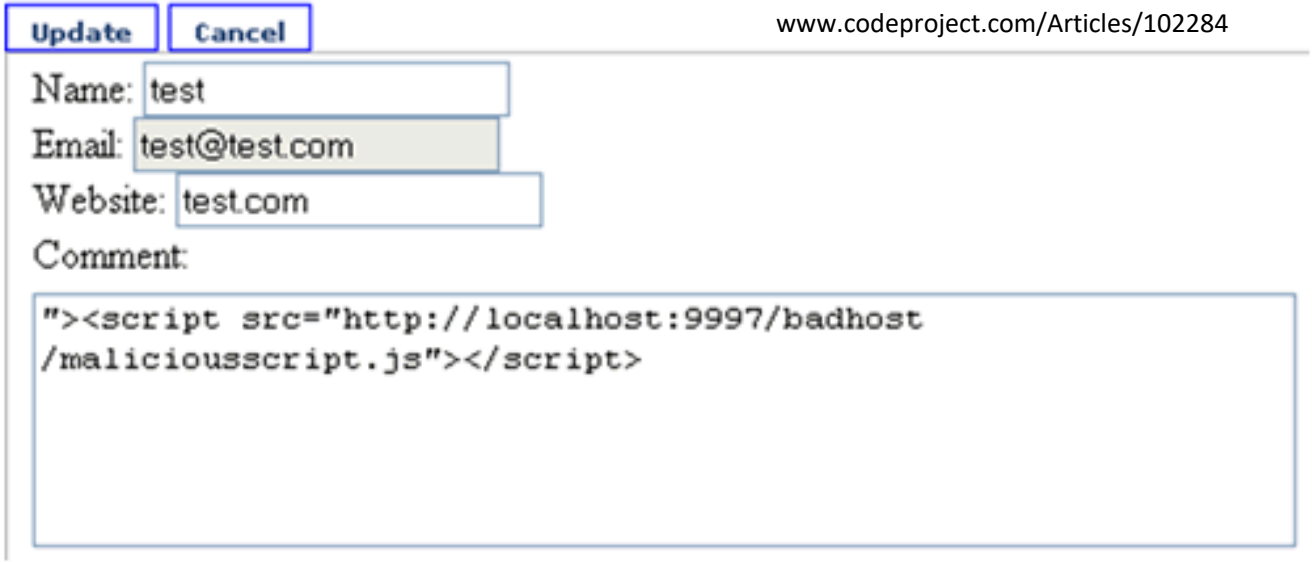
OWASP Mobile Top 10



OWASP
Mobile Security Project

Injection Flaws - XSS

- Cross Site Scripting (XSS)
  - injecting code into a Javascript-enabled WebView and causes it to behave differently than it should be
  - e.g. inject a key logger that grab login/passwords back to the attacker



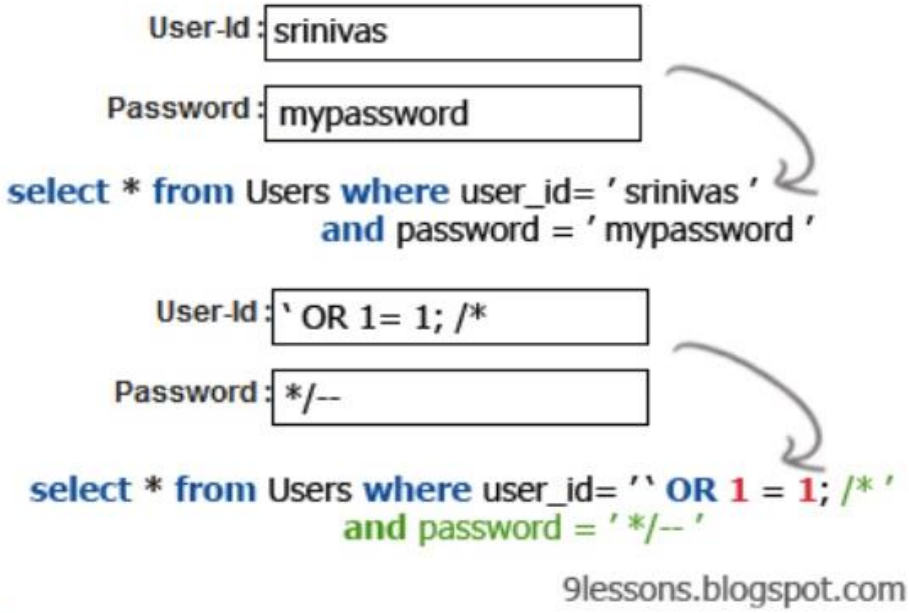
Combating XSS

Preventing XSS is complicated…

- In Android, if **WebView** does **not execute JavaScript**, no cross-site-scripting is possible
- So, if your application doesn't directly use JavaScript within a **WebView**, do not call **setJavaScriptEnabled()**, or,
- Treat HTML document like a parameterized database query
  - keep data in specific places, and
  - isolated data from code contexts with escaping:

Injection Flaws - SQLi

- SQL injection
  - a technique that takes advantage of the **syntax of SQL** to **inject arbitrary commands**
  - e.g. Inject malicious queries to an SQL database to grant access



supplementary

Combating SQLi

By use of parameterized queries to SQL database

- In Android, queries to SQL DB or content provider, should be done through `query()`, `update()`, and `delete()` etc.
- In iOS, can use special *Formatting String Objects* such as Predicate to avoid naive SQL commands

Insecure Data Storage

The most common security concern in Android:

- whether the data that you save on the device is accessible to other apps

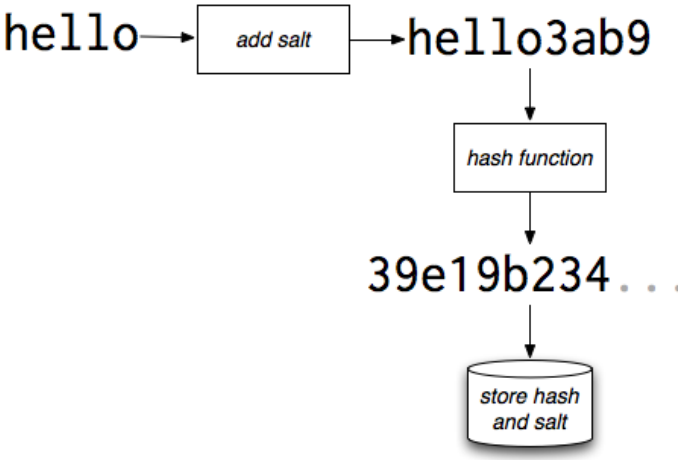

```
public void saveCredentials(String userName, String password) {

    SharedPreferences credentials = this.getSharedPreferences(
            "credentials", MODE_WORLD_READABLE);   — Very Bad
    SharedPreferences.Editor editor = credentials.edit();
    editor.putString("username", userName);
    editor.putString("password", password);   — Convenient!
    editor.putBoolean("remember", true);
    editor.commit();
}
```

Securing Data

- Handling credentials with **authorization token** instead of storing <u>usernames</u> and <u>passwords</u> on the device e.g., **AccountManager.getAuthToken()**

- Use Salt in hashing password



- Using EncryptedSharedPreferences
  - Values are encrypted using AES-256 GCM

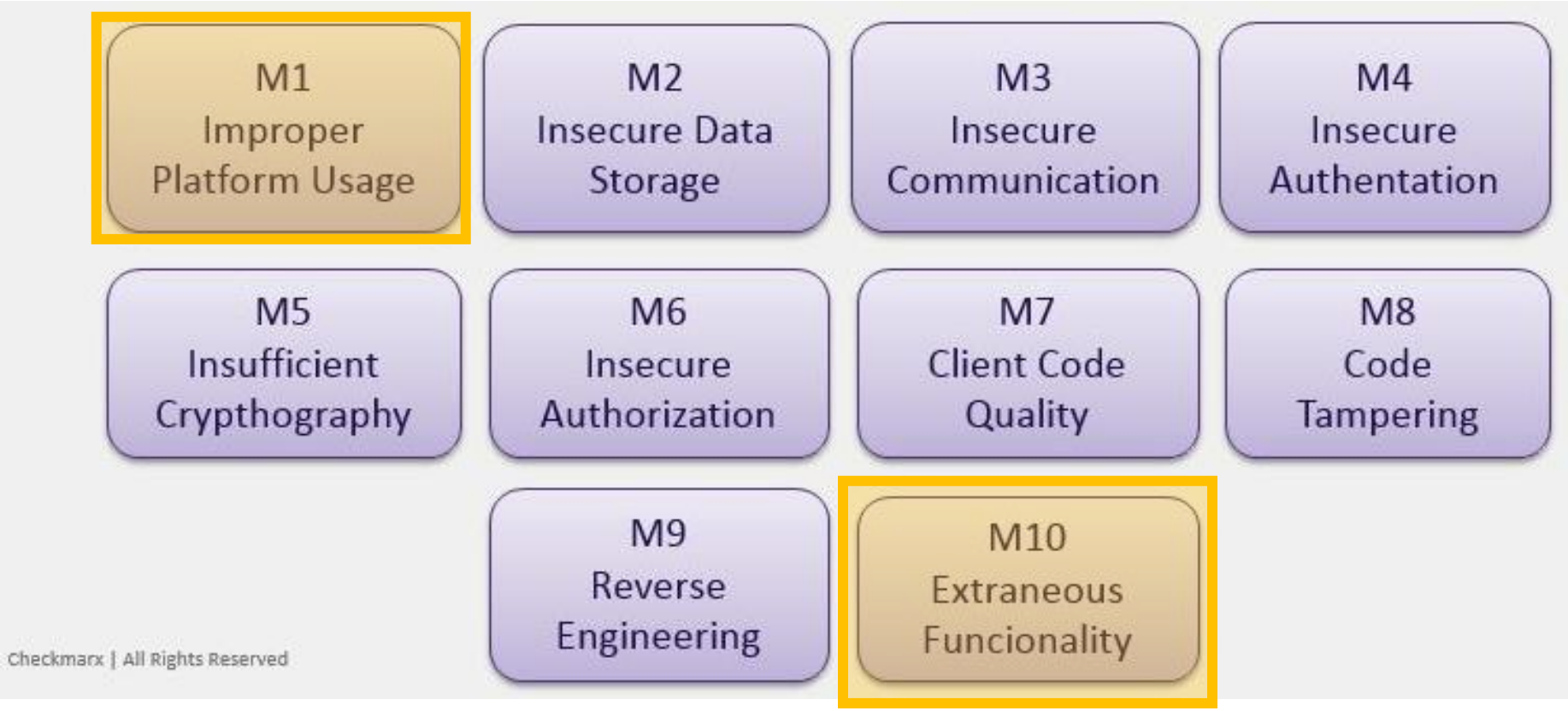
Rules for Handling Input Data

We always need to validate the following items in any code that handles data from an untrusted source.

- (a) Does the received data **match the format** that was expected by the programmer and does the value fall in the expected scope?

- (b) Even if you have received the expected format and value, can you guarantee that the code which handles that data will not **behave unexpectedly?**

OWASP Mobile Top 10



OWASP
Mobile Security Project

Android OS vulnerabilities

- Liblog
  - Somewhat buggy…
  - Log devices are world writeable (/dev/log/*).
  - Arbitrary log writing possible.
- Logcat uses liblog
- Logcat instances can be exploited to disable log monitoring functionality in many apps. *Code execution may be possible* due to nature of vulnerabilities (**heap corruption**).
- Possibility exists of exploiting the logging vulnerabilities remotely due to nature of vulnerabilities. Similar bugs found in library previously

Remove all sensitive **Logcat** statements in **deployment**

Architecture vulnerabilities

- Manifest & Permissions
  - Designed to let Android apps to declare the access needed
- All required access will be displayed to user during installation
- Supposedly user will be informed / have control over the app
- Developers will also be constrained
- But main point is it put average consumer in charge of the critical security decision making

Permission Use

- Compared number of permissions requested in 1,400 legit apps vs. 760 malicious apps [†]
  - Range was as high as 39 for a malicious app and 34 for a legit app (NetQin Mobile Anti-virus)
  - Median number of permissions: **7 for malicious**, **3 for legitimate**
  - More users now **pay attention when they install them**

[†] **Sourcefire Vulnerability Research Team** (VRT) http://vrt-blog.snort.org/

Manifest & Permissions

CALL_PHONE

- Allows an application to **initiate a phone call without** going through the **Dialer user interface** for the user to confirm the call being placed.
- Will show a dialog box to user requesting for phone call be initiated

- Typically requested by many apps even they didn't use it
- A usual pattern for many apps – requests many permissions but don't actually use it

Risky Permissions

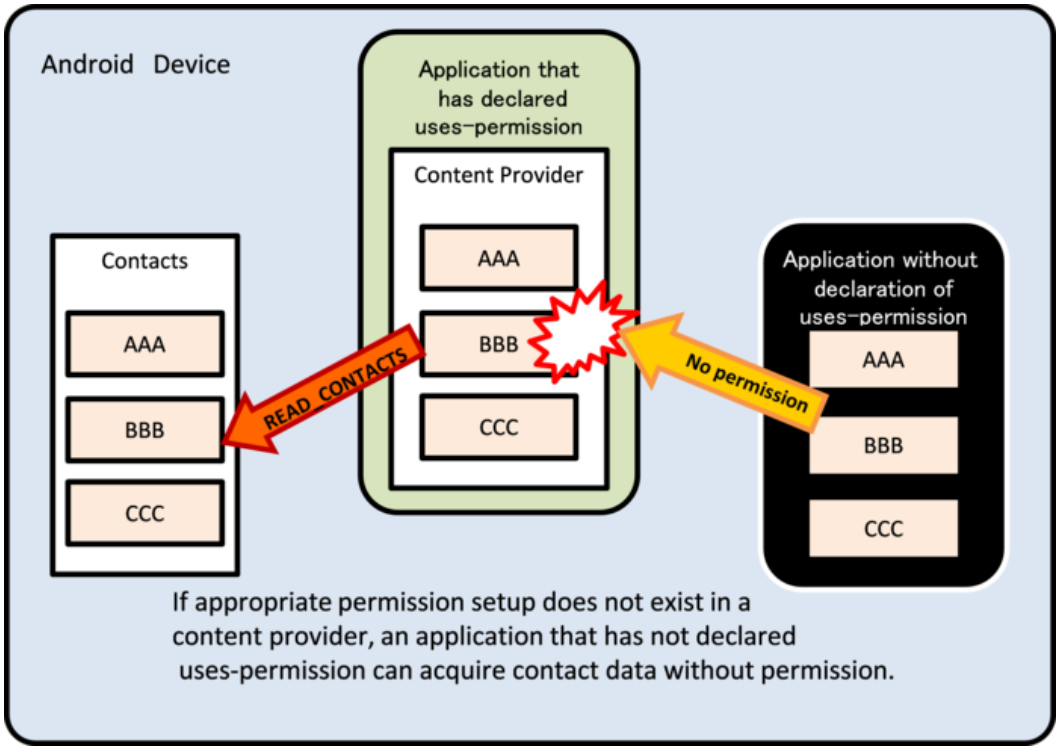
SEND_SMS

- Happens completely in background, no dialog box as in CALL_PHONE
- **Text message can be charged instantly** …

- As a developer, **minimize access to sensitive permissions**…
  - not only **improves user adoption**,
  - but also makes your app **less vulnerable for attackers**.

Permission Re-delegation Problem

- Android's permission mechanism is only able to manage permission of direct access from original app to protected data.
- As a developer, **do not expose unnecessary permission to other** applications, e.g., through *ContentProvider*



Reference

1. **OWASP Mobile Security Project**
   https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
   https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

2. **Android Application Secure Design/Secure Coding Guidebook** (JSSEC)
   https://www.jssec.org/dl/android_securecoding_en/

3. **Best Practices for Security & Privacy** (Android Developer Portal)
   https://developer.android.com/training/safetynet/attestation
   https://developer.android.com/training/articles/security-tips.html
   https://developer.android.com/training/articles/security-gms-provider

(Supplementary)
Mobile Device Management

supplementary

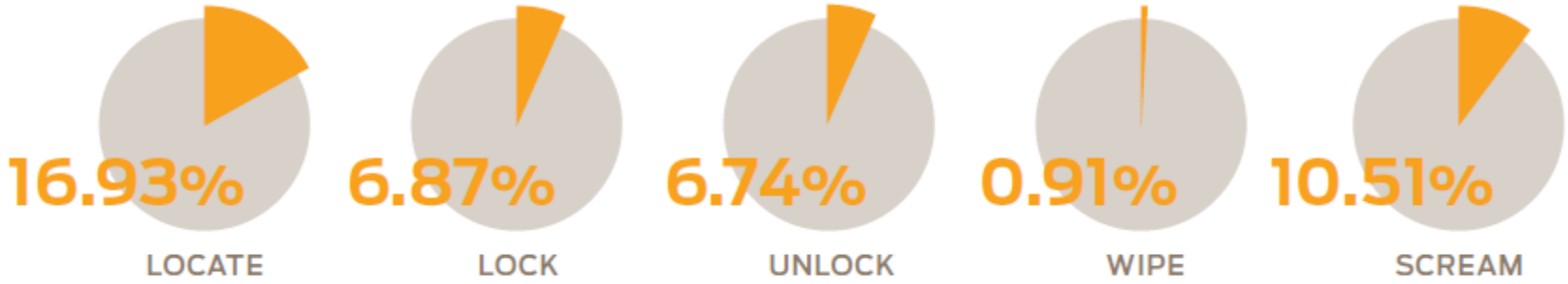
Managing Lost Device

- Mobile device management concerns on mitigating damage caused by a lost or stolen device, especially when the device carries **sensitive corporate or personal information**

- Risks including
  - Data breach : lost or stolen device with customer or employee information carry legal and reputational costs
  - Loss of intellectual property and trade secrets : IP when falls on wrong hands could have devastating effects on business
  - Loss of personal information : stolen personal information could be used for malicious purposes such as fraud and identity theft

Managing Lost Device

- Services available <u>to reduce/recover loss after the device being taken away</u>
- Commands such as "**locate**", "**lock**", "**unlock**", "**wipe** (most damaging)", and "**scream** – cause a loud audible tone to emanate from a device within vicinity
- Low rate of "**wipe**" indicated that other commands can lead to device recovery



supplementary

Device Admin for Developer

- Enterprise now use Mobile Device Management (**MDM**) software to manage employee mobile devices
  - Policy enforcement
  - Remote wipe
  - Device locating
- Connect
  - SSL VPN client to protect data in transit
  - Network access control based on user identity and device security posture
- Defend
  - Centralized remote locate, track, lock, wipe, backup and restore facilities
  - On-device host checking to assess OS version, malware/rooted status

Both Android and iOS provide **MDM** API:

https://developer.android.com/guide/topics/admin/device-admin.html

https://developer.apple.com/library/content/documentation/Miscellaneous/Reference/MobileDeviceManagementProtocolRef

supplementary