Module 3: Android Vulnerabilities

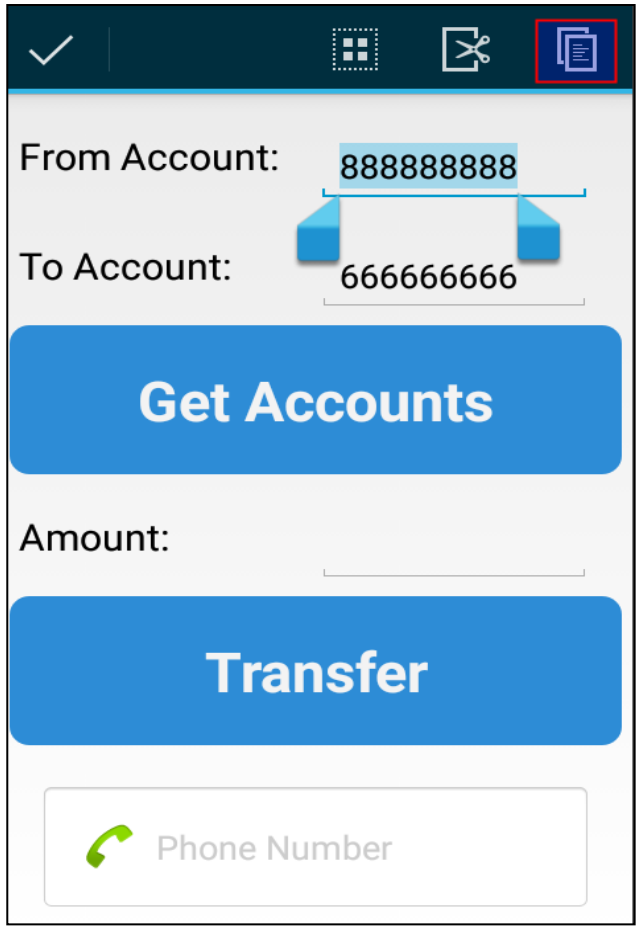
- Exploiting Local Storage
- Exploiting Android Components
- Exploiting Weak Cryptography
- Exploiting Side Channel Data Leakage
- Root Detection and Bypass
- Exploiting Network Communication and Cert Pinning
- Exploiting Firebase Databases
- Attacking Google Play Billing

Exploiting Side Channel Data Leakage

- Vulnerabilities Include:
  - Copy Paste Buffer
  - Android Keyboard Cache
  - Android Log Files

Exploiting Side Channel Data Leakage Copy Paste Buffer

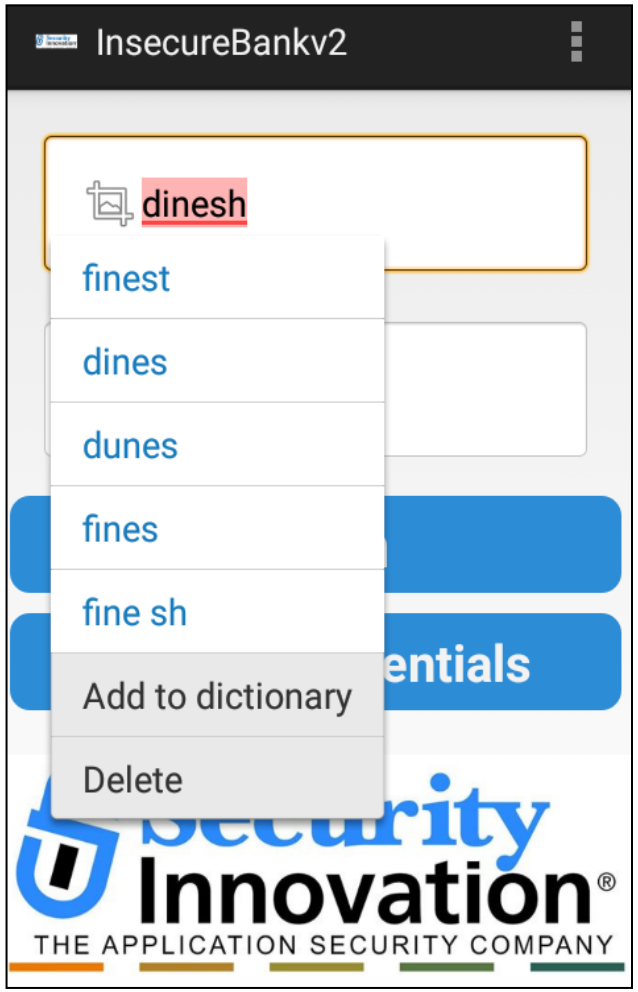
- Android has a common copy-paste buffer for all the applications
  - Data in this buffer is shared across different applications
- When copy/paste is allowed on sensitive fields like credit card numbers, SSN, usernames, etc., the data in the buffer is available to other applications
  - This could result in the leakage of sensitive information
- The copied text is stored locally




```
adb shell ps | grep insecure
adb shell su u0_a58 service call clipboard 2 s16
com.android.insecurebankv2
```

Exploiting Side Channel Data Leakage Android Keyboard Cache

- Keystrokes are logged to help with auto-correction and form completion
- New words are cached as part of the Android auto-correction feature
  - It is turned on by default for all textfields except password fields
- When auto correction is allowed on sensitive fields like credit card numbers, SSN, usernames, etc., this could allow sensitive information to be stored as keywords on the device
- The keywords are stored locally at /data/data/com.android.providers.userdictionary/databases/user_dict.db



Exploiting Side Channel Data Leakage Android Log Files

- Android maintains a centralized logcat of all the apps and device logs
- Android uses circular buffers for log messages
  - adb logcat [-b buffer_name] dumps the logs to stdout
    - main is the default buffer
    - radio contains radio/telephony related messages
    - events contains events-related messages
    - The –f <filename> parameter dumps the buffer to <filename>
- Can be accessed with READ_LOGS permission before 4.1
- Logcat permission has now changed to signaturelsystemI development

HOMEWORK

- Go through https://www.exploit-db.com/exploits/46933
- Go through https://www.exploit-db.com/exploits/39061

Module 3: Android Vulnerabilities

- Exploiting Local Storage
- Exploiting Android Components
- Exploiting Weak Cryptography
- Exploiting Side Channel Data Leakage
- Root Detection and Bypass
- Exploiting Network Communication and Cert Pinning
- Exploiting Firebase Databases
- Attacking Google Play Billing

Root Detection and Bypass
Detecting a Rooted Device

- Developers may want to detect if a device has been rooted for a variety of reasons
  - For example, they may not want the application to be run on a rooted device
- There are many ways to detect a rooted device
  - Default files and configurations that are found on rooted devices (not able to access on non-rooted devices due to permissions)
    - Release-keys tags, Over the Air (OTA) certificates
  - Installed files and packages present on rooted devices
    - Superuser.apk, su
  - Changed directory permissions
    - Rooted devices will have certain directories that are writable
    - /data is readable
  - If certain commands run, that can be an indication of a rooted device
- Not all methods work consistently
  - Every developer has their own “secret sauce” for detected a rooted device

Root Detection and Bypass
Detecting a Rooted Device

```
private boolean doesSUexist() {
    Process process = null;
    try {
        process = Runtime.getRuntime().exec(new String[] { "/system/xbin/which", "su" });
        BufferedReader in = new BufferedReader(new InputStreamReader(process.getInputStream()));
        if (in.readLine() != null) return true;
        return false;
    } catch (Throwable t) {
        return false;
    } finally {
        if (process != null) process.destroy();
    }

}
```

Root Detection and Bypass
Detecting a Rooted Device

```
private boolean doesSuperuserApkExist(String s) {

    File rootFile = new File("/system/app/Superuser.apk");
    Boolean doesexist = rootFile.exists();
    if(doesexist == true)
    {
        return(true);
    }
    else
    {
        return(false);
    }
}
```

Root Detection and Bypass
Detecting a Rooted Device

- Check for the BUILD tag
  - `cat /system/build.prop | grep ro.build.tags`
    - ro.build.tags=release-keys

- Check for Over The Air (OTA) certificates
  - `ls -l /etc/security/otacerts.zip`
    - -rw-r--r-- root root 1125 2020-07-26 15:29 otacerts.zip

Source: https://www.netspi.com/blog/entryid/209/android-root-detection-techniques

Root Detection and Bypass
Bypassing Root Detection

```
me$  adb  shell
$  su
# cd /system/bin/; mount -o remount,rw -o rootfs rootfs /; mount -o
remount,rw -o yaffs2 /dev/block/mtdblock3 /system
# echo $PATH  /sbin:/system/sbin:/system/bin:/system/xbin
# mv /system/sbin/su  /system/xbin/
```

Source: http://www.floyd.ch/download/Android_0sec.pdf

Root Detection and Bypass
Bypassing Root Detection

```
me$ java -jar apktool.jar d  app.apk  source
[...]
me$  sed  -i "" 's/systemV sbin V su /systemV sbin V CEW1PFSLK/
g' source/ smali /net/example/ checks.smali
me$ java -jar apktool.jar b source/  fake.apk [...]
me$  keytool  - genkey  -alias someone -validity 30  - keystore
someone.keystore [...]
me$  jarsigner  - keystore someone.keystore fake.apk someone
me$  adb  install  fake.apk
```

Source: http://www.floyd.ch/download/Android_0sec.pdf