# Decrypting Crypto POS

## A Fun Dive into this Hacking Playground

Guanxing Wen

# $ whoami

- Security Researcher @ CertiK
- Former core member @ Pangu Team
- Big fan of Bootloader, Trustzone and Pwning smart devices, 70+ CVEs
- 1st place of Huawei Bug Bounty Program in 2021 & 2022

# POS (Point of Sale)


Strip / Chip / NFC

POS Terminal


Card Networks


Card Issuing Banks

Linux, Windows, Custom OS

Android Based


1980s - 2010s

Now

# From Silicon to Service

- Chipset Vendor Collaboration
  - Vendors like MediaTek or Qualcomm
  - Produce workable bootloader, Linux kernel, and AOSP

# From Silicon to Service

- POS Terminal Manufacturing
  - Manufacturers like PAX, Clover, Newland, Landi, etc.
  - Hardware — strip + chip + nfc card reader and secure element
  - Software — specific drivers & card network related applets
  - Enhancing Security Mitigations (anti-root, anti-tamper)

# From Silicon to Service

- Payment Companies Customization
  - Companies like Lakala, Unionpay, Shopify, Helcim, etc.
  - Additional branding elements (logo, color theme, splash screen)
  - Customized applications tailored to restaurants, hotels, stores, etc.

# Hack in Action

# Hack in Action

- Pwned at GeekPwn 2017
- APOS A8, Android 5.1.1
- Qualcomm + Landi + Lakala
- Starts with bluetooth

- Ends up with duplicate bank cards for fraudulent transactions

# Hack in Action

- Transfer APK into the device via Bluetooth and proceed with installation
  - ✖ Network verification
  - ✔ Use WiFi-proxy to hijack connections

```
{errorMessage: xxxx errorCode: 0000}
```

# Hack in Action

- epay.sysmgrhelper grants for unrestricted writing capabilities
  - Accept an ini filename as parameter

```
[storagepath] = /sys/kernel
[storagename] = uevent_helper
[specialfiledata] = /var/.download/payload
```

# Hack in Action

- /var/.download/payload executes as u:r:kernel:s0 upon user PIN input
  - Retrieve PIN cache through **/dev/kmem**

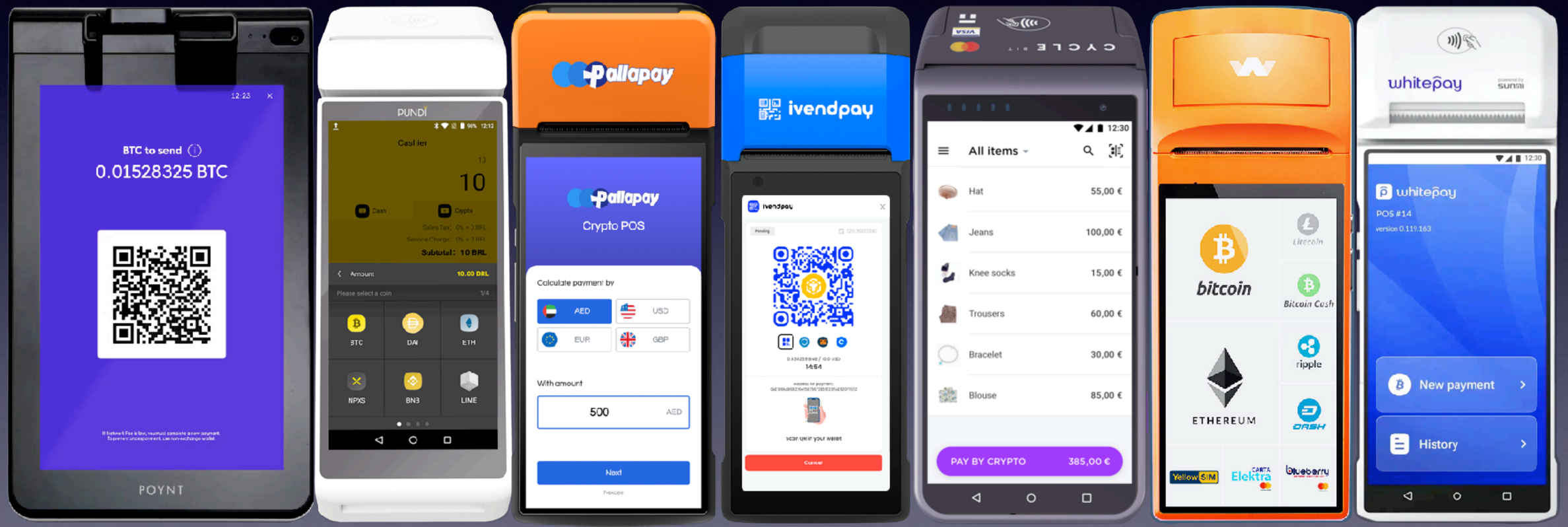
```
if ( pinKeyCode >= '0' )
{
  v5 = pThis->inputedPinLen;
  if ( pThis->maxPinLen > v5 )
  {
    pThis->clrPinBuf[v5] = pinKeyCode;
    v11 = (unsigned __int8)(v5 + 1);
    pThis->inputedPinLen = v11;
    memcpy(&clrPin_25988, pThis->clrPinBuf, v11);
    v12 = v3->inputedPinLen;
    v3->pinEntryInfo.mState = 0;
    v3->pinEntryInfo.mPinNumInputed = v12;
    *((_BYTE *)&pin_catcher_data + v12 + 984) = 0;
    v3->pinEntryInfo.mKeyCode = '*';
    if ( v3->pinEntryCfg.mPinLenTypes != 1 || v3->pinEntryCfg.mPinLenTypesList[0] != v12 )
      goto LABEL_11;
```

# Hack in Action

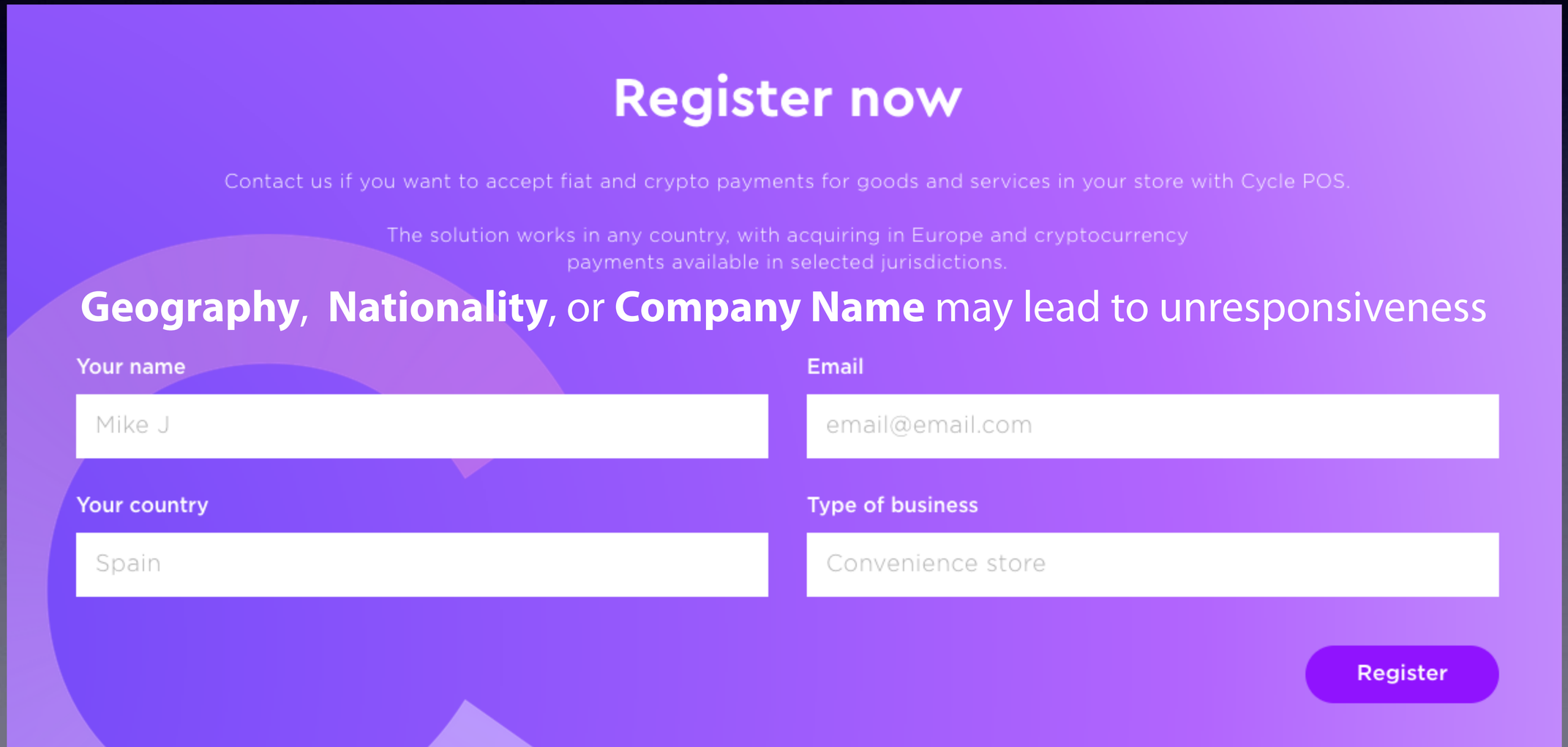
- Repack com.lkl.cloudpos.payment
  - Disable card type verification
  - Transmit card magnetic stripes and PIN to my server

# DEMO

# New Era: Crypto POS



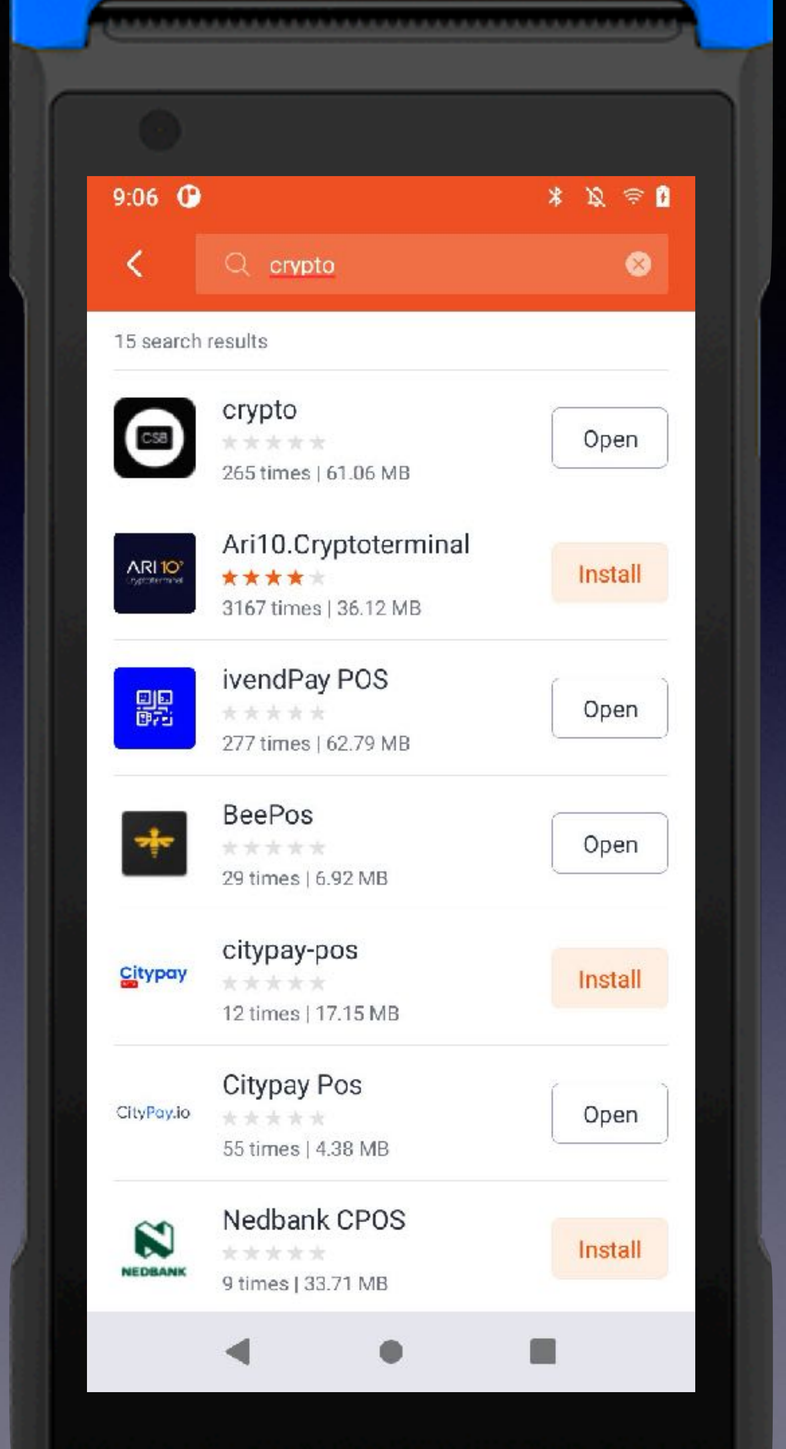
# Acquiring Devices is Challenging



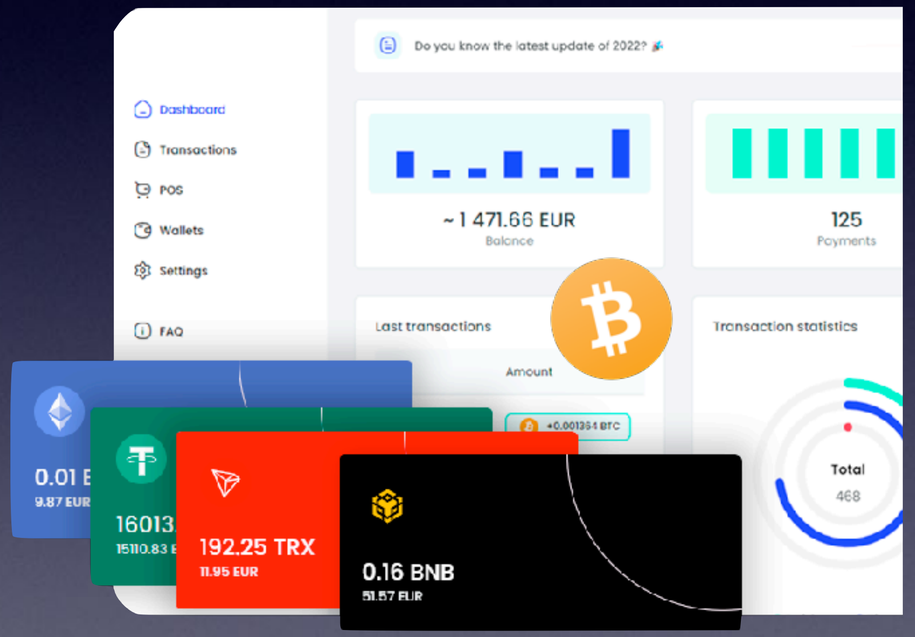
# Acquiring Devices is Challenging

- Chipset Vendors -> **Terminal Manufactures** -> Payment Companies
- Terminal Manufactures ≈ Payment Companies
  - Very limited customizations from Payment Companies
  - Minimal need for TEE/Secure Elements or extra hardwares
- **Wide Impact on Crypto and Traditional POS**

# Inner Workings



# Inner Workings




# “Crypto POS Terminals”

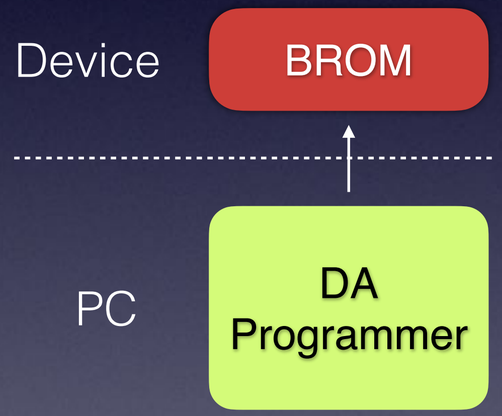
- Acquired multiple devices covering Android 6, 7, 11
- Less customization than traditional smart POS
  - Selinux is enforced
  - Third-party APKs is not allowed
  - ADB is bound to specific developer account
- **We are also much stronger than before**

# Starting Point

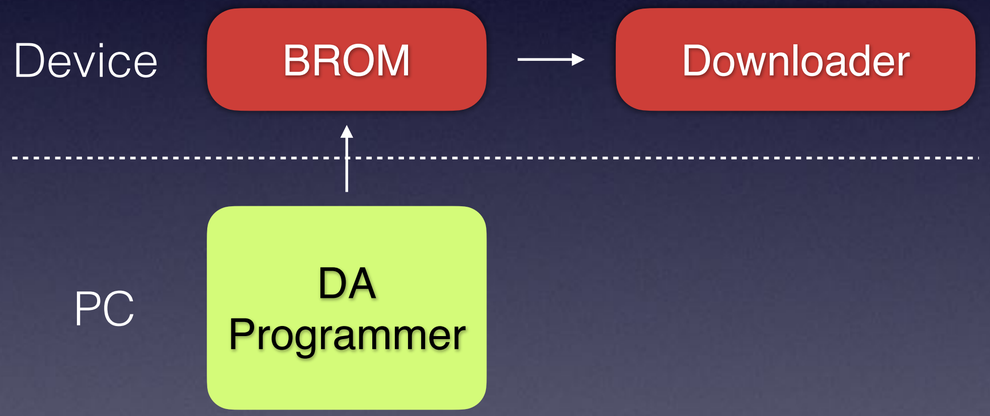
- Bootchain provides us with new attack surface
  - USB Download mode is available for mainstream chipsets
    - Qualcomm — **alephsecurity**.com/2018/01/22/qualcomm-edl-1/
    - Mediatek — github.com/**bkerler**/mtkclient
    - Exynos — github.com/**frederic**/exynos-usbdl
    - Kirin — github.com/**hhj4ck**/checkm30

# USB Download Mode

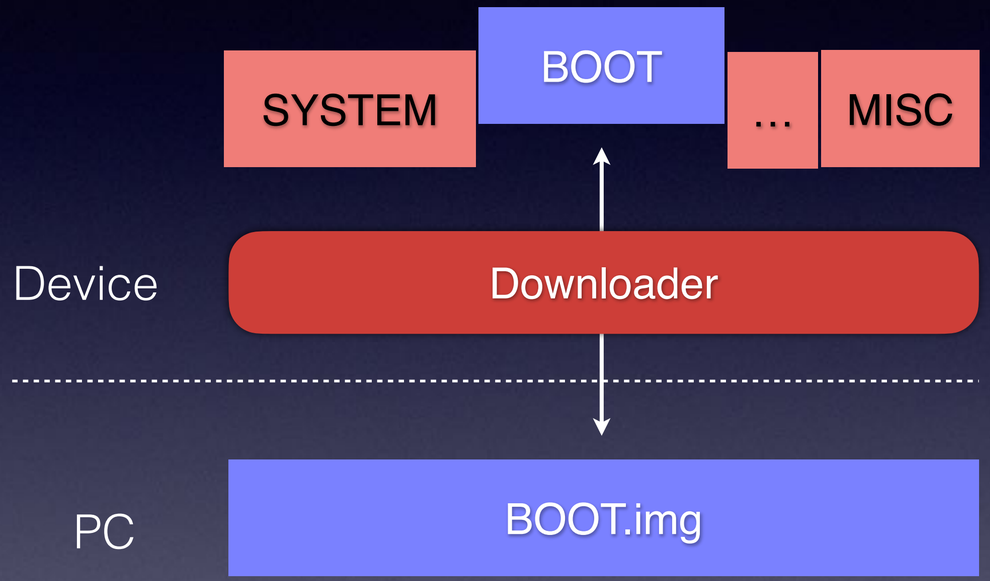
# USB Download Mode



# USB Download Mode

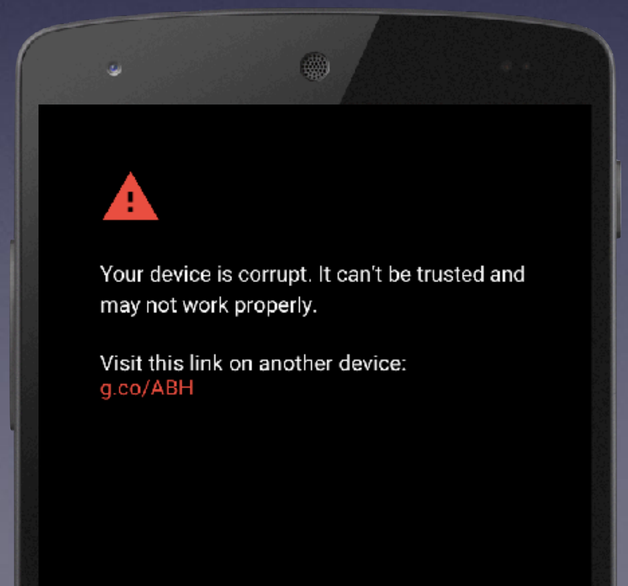


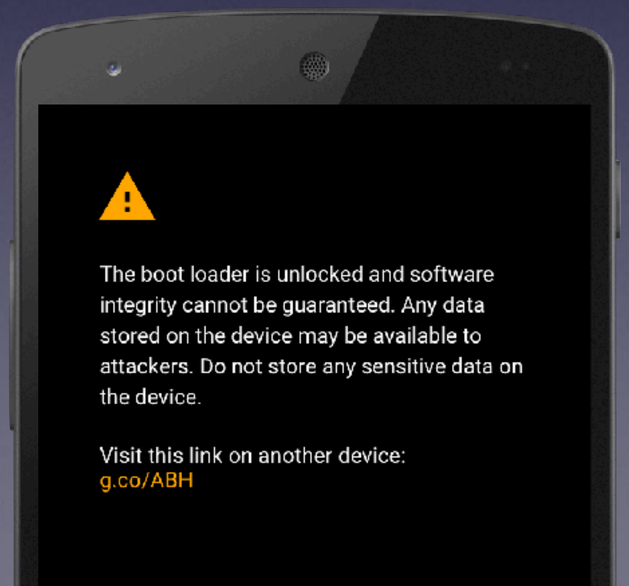
# Partition Read/Write

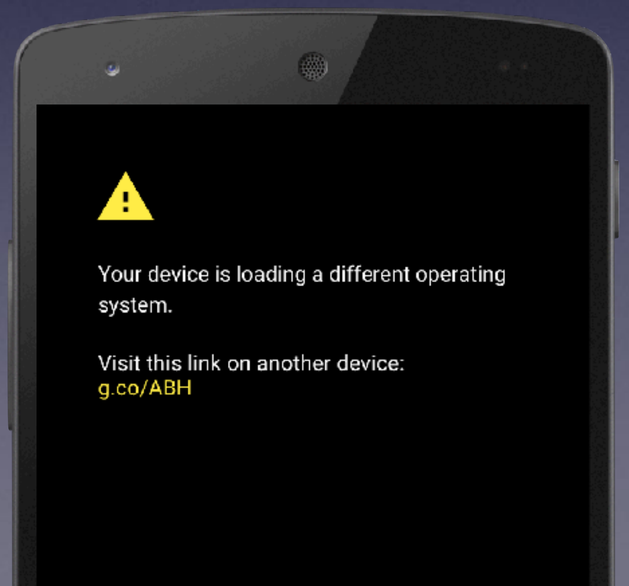


# Normal Boot

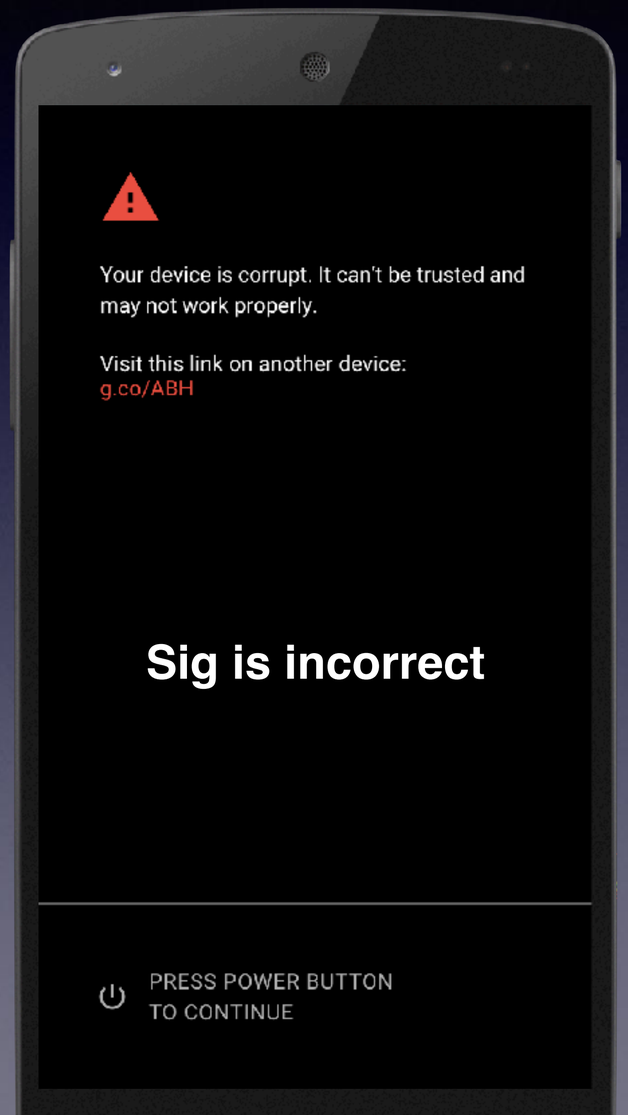
# Bootloader State

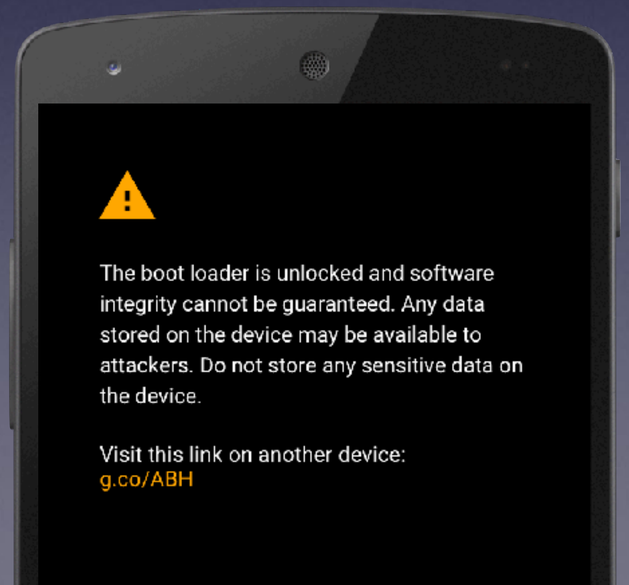


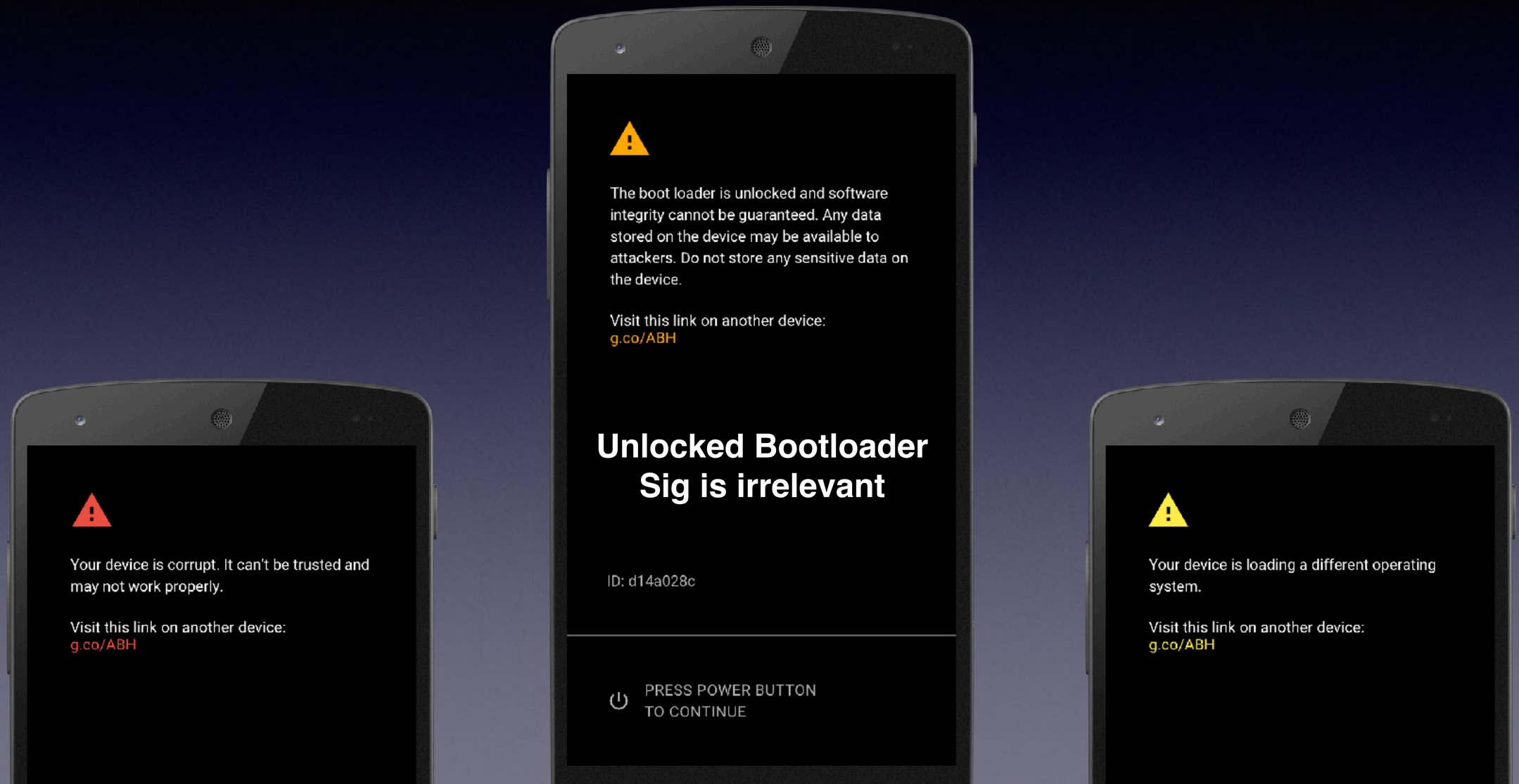



# Bootloader State

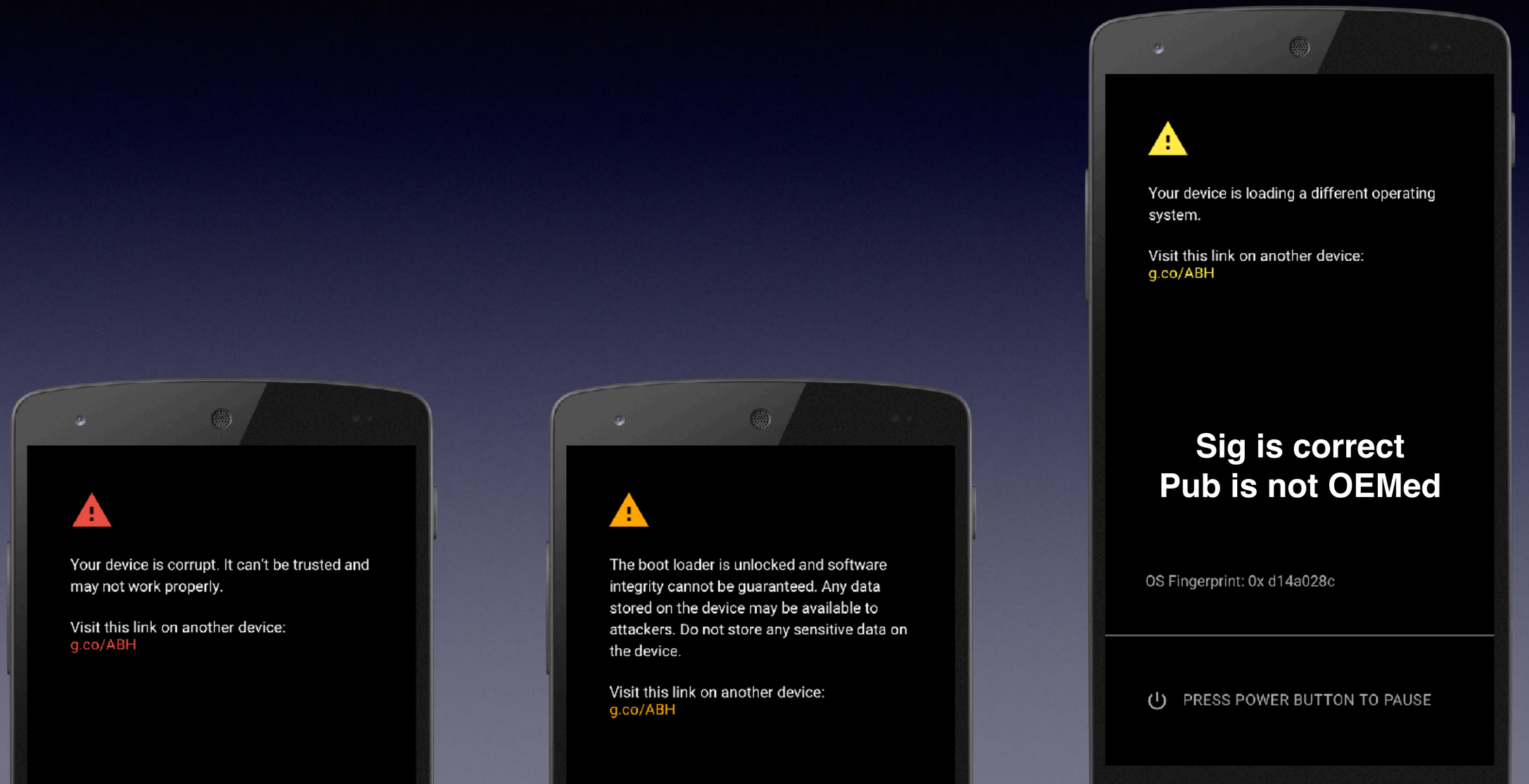



# Bootloader State



# Bootloader State



# Android Booting Shenanigans

- Verifying boot.img is not an easy task
- So many influencing factors in history
  - AVB 1.0, AVB 2.0, A/B slots, boot as recovery, Project Treble …
  - Bootloader takes tremendous considerations from all of them
  - **Inconsistencies may appear here or there**

# Case One

- Android 6, AVB 1.0, MediaTeK
  - SBC is not correctly configured
    - sec_policy->boot_chk_bootimg = 0
    - Verification of boot.img is **bypassed**

# Case Two

- Android 7, AVB 1.0, Qualcomm
  - The Sig and Pub is attached at the end of boot.img
  - Bootloader compares the Pub with an OEMed certificate
    - What if the Pub is not OEMed?

# Case Two: Yellow state is allowed

```
v24 = (_DWORD *)X509_get_pubkey(sig[1]);
if ( !v24 )
{
  dprintf(aUnableToExtrac);
  ASSERT(v3, "ASSERT FAILED at (%s:%d): %s\n", aPlatformMsmSha_11, 324, &aCardSizeSec0[16]);
}
dword_8F6A4280 = EVP_PKEY_get1_RSA(v24);
v17 = boot_verify_compare_sha256(img_addr, _img_size, *(_DWORD *)(sig[4] + 8), dword_8F6A4280);
if ( v17 )
  boot_verify_send_event(2);   // BOOTIMG_EMBEDDED_CERT_VERIFICATION_PASS = 2
else                           // BOOTIMG_KEYSTORE_VERIFICATION_PASS = 3
  sub_8F60FE14();
```

```
char *__fastcall display_bootverify_menu_renew(int *msg_info, int type)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v10 = stack_cookie;
  if ( type == 8 )
  {
    fbcon_clear();
    memset(msg_info, 0, 84);
    v7 = aIfPowerKeyIsNo;
    str = (char *)'\n';
    scale_factor = common_factor;
    do                                          // display_fbcon_menu_message
    {
      fbcon_putc_factor(str, 0, scale_factor);
      v9 = (char *)(unsigned __int8)*++v7;
      str = v9;
    }
    while ( v9 );
    v5 = 30000;
  }
  else
  {
    str = (char *)memset(msg_info, 0, 84);
    v5 = 1;
  }
  v6 = stack_cookie;
  msg_info[17] = type;
  msg_info[18] = v5;
  if ( v10 != v6 )
    corrupt_cookie();
  return str;
}
```

// DISPLAY_MENU_YELLOW = 0,
// DISPLAY_MENU_ORANGE = 1,
// DISPLAY_MENU_RED = 2,
// DISPLAY_MENU_EIO = 8

**Yellow Warning is dismissed!!!**

# Case Three

- Android 11, AVB 2.0, MediaTek
  - Yellow state is not allowed
  - Sig and Pub of boot.img lies within its footer
  - Bootloader compares the boot's Pub with the Pub from vbmeta
  - Bootloader verifies the Pub of vbmeta using a hardcoded Pub key

```
int __fastcall avb_hal_verify_public_key(
        int ops,
        unsigned __int8 *public_key_data,
        int public_key_length,
        int public_key_metadata,
        int public_key_metadata_length,
        _DWORD *out_is_trusted)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v6 = public_key_data == 0;
  if ( public_key_data )
    v6 = out_is_trusted == 0;
  if ( v6 )
    return 2;
  v9 = avb_htobe32(*(_DWORD *)public_key_data);
  *out_is_trusted = 0;
  if ( v9 >> 3 == 256 )
  {
    pubk = public_key_data + 8;
    if ( !sec_set_pubk((int)pubk, 256) )
    {480BC170 g_avb_key       DCB 0xC6, 0x55, 0x51, 0xDD, 0x32, 0x24, 0xA2, 0xE0, 0xE
      result = memcmp(g_avb_key, pubk, 256);
```

# Case Three: g_avb_key is leaked

```
$ openssl rsa -in leaked.pem -text

  Private-Key: (2048 bit, 2 primes)
  modulus:
      00:c6:55:51:dd:32:24:a2:e0:0e:bc:7e:fd:bd:a2:
      53:80:58:69:7e:f5:4a:40:87:95:90:54:59:3d:55:
      ca:ff:36:34:1a:fa:e1:e0:90:2a:1a:32:68:5b:f3:
      df:ad:0b:f9:b1:d0:f7:ea:ab:47:1f:76:be:1b:98:
      4b:67:a3:62:fa:df:e6:b5:f8:ee:73:16:5f:b8:b1:
      82:de:49:89:d5:3d:d7:a8:42:99:81:75:c8:d8:84:
      7b:bd:54:a8:22:64:44:bc:34:06:10:3c:89:c2:d1:
      ...
  publicExponent: 65537 (0x10001)
  privateExponent:
      00:be:90:98:1a:0a:e5:fc:c1:80:27:49:75:90:d1:
```

# Hack in Action

# Case Three: Persistent Root

1. Extract and replace the Pub of boot.img in vbmeta.img

```
$ ./avbtool extract_public_key --key our_boot.pem --output our_boot.pub ...
$ ./avbtool make_vbmeta_image --key leaked.pem --chain_parition ...
```

# Case Three: Persistent Root

1. Extract and replace the Pub of boot.img in vbmeta.img
2. Repack boot.img (backdoor init) and re-sign

```
$ ./avbtool add_hash_footer --image boot.img --key our_boot.pem ...
```

# Case Three: Persistent Root

1. Extract and replace the Pub of boot.img in vbmeta.img
2. Repack boot.img (backdoor init) and re-sign
3. Overwrite boot partition with usb download

# Case Three: Persistent Root

1. Extract and replace the Pub of boot.img in vbmeta.img
2. Repack boot.img (backdoor init) and re-sign
3. Overwrite boot partition with usb download
4. Remote root shell connects back whenever the device is on

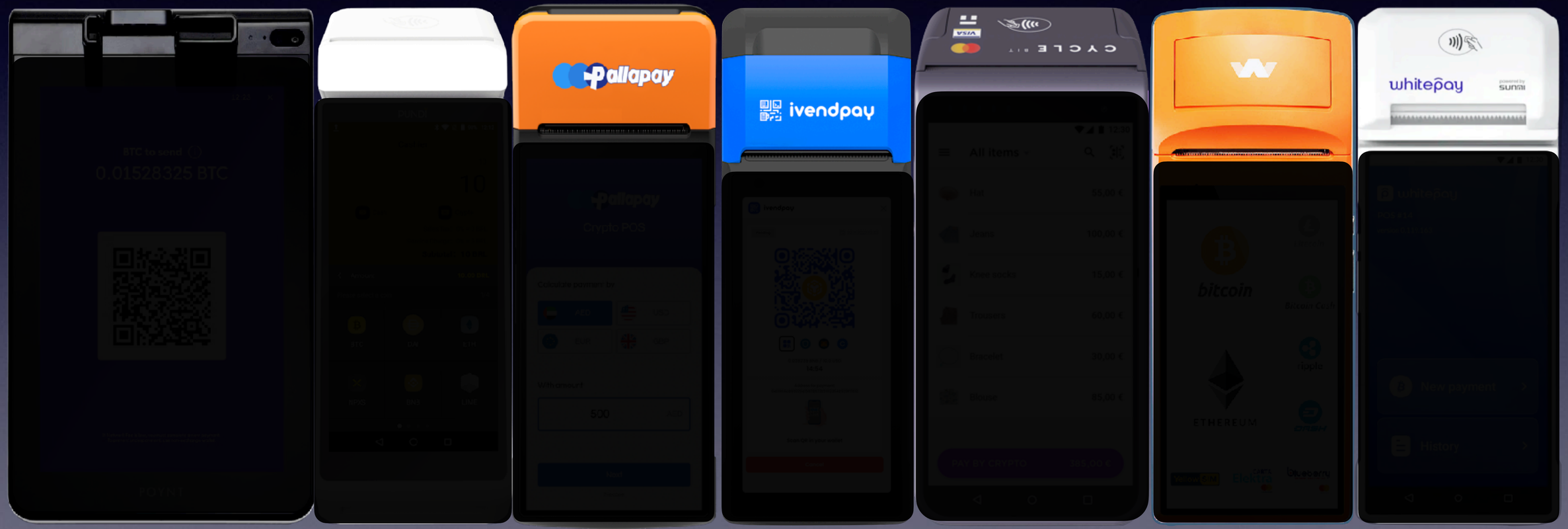
# Case Three: Persistent Root

1. Extract and replace the Pub of boot.img in vbmeta.img
2. Repack boot.img (backdoor init) and re-sign
3. Overwrite boot partition with usb download
4. Remote root shell connects back whenever the device is on

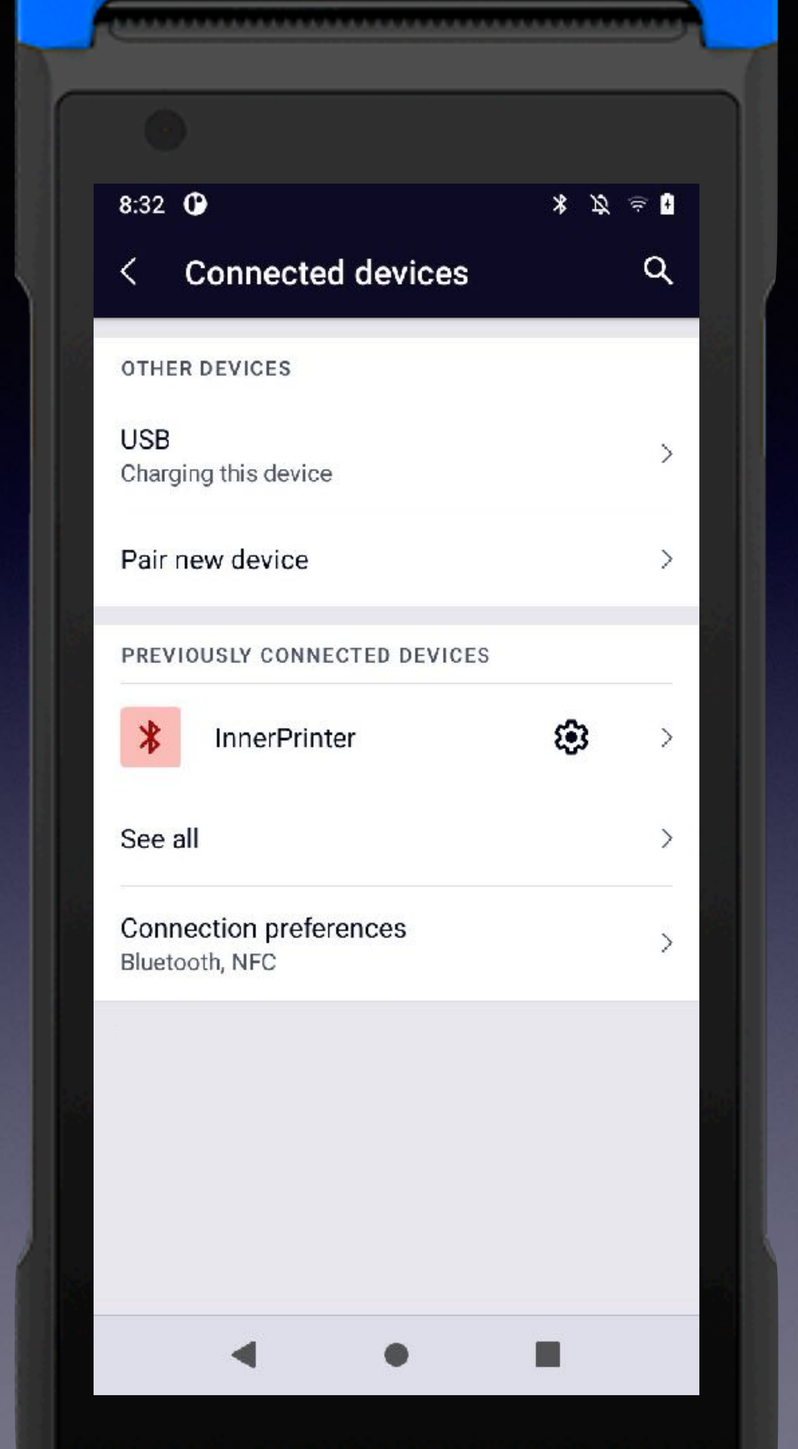
   Replace seller’s session token, forthcoming $$$ will be sent to us

# Demo

# POS Printers



# Virtual Bluetooth Printer



# Virtual Bluetooth Printer

```
void clientConnect(int clientIf, String address, boolean isDirect, int transport,
    enforceCallingOrSelfPermission(ProfileService.BLUETOOTH_PERM, "Need BLUETOOTH
    if (DBG) {
        Log.d(TAG, "clientConnect() - address=" + address + ", isDirect=" + isDire
    }
    if (SimulateDevicesUtils.SIMULATE_BLE_IMAGING_ADDRESS.equals(address)) {
        try {
            onConnected(clientIf, -1, 0, address);
            return;
        } catch (RemoteException e) {
            Log.w(TAG, "err to trigger onConnected", e);
        }
    }
    gattClientConnectNativeNative(clientIf, address, isDirect, transport, opportunistic,
}
```

```
private static final int ET_CONNECTABLE_MASK = 1;
private static final int ET_LEGACY_MASK = 16;
public static final int SIMULATE_BLE_ADDRESS_TYPE = 0;
public static final int SIMULATE_BLE_ADVERTISING_SID = 255;
public static final String SIMULATE_BLE_CHARACTERISTIC_UUID = "00001101-0000-1000-8000-00805F9B34FB";
public static final int SIMULATE_BLE_CONNECTION_ID_PLUS = 1000;
public static final int SIMULATE_BLE_EVENT_TYPE = 17;
public static final String SIMULATE_BLE_IMAGING_ADDRESS = "00:11:22:33:44:55";
```

# Virtual Bluetooth Printer

- Virtual Bluetooth Device - 00:11:22:33:44:55
  - Hook GattService::{ClientConnect, OnConnected, ClientDisconnect, discoverServices}
- ESP/POS
  - A variant of  Epson Standard Code for Printers for controlling receipt printers

```
public class ServerFactory {
    public static boolean SIMULATE_IMAGING_ENABLE = SimulateDevicesUtils.SIMULATE_IMAGING_ENABLE;
    public static boolean BT_IMAGING_ENABLE = true;

    public static void newServers(Context context, ExecutorService executorService, Printer printer,
        if (list != null) {
            list.clear();
            if (SIMULATE_IMAGING_ENABLE) {
                list.add(new SimulateSocketServer(context, executorService, printer));
            }
            if (BT_IMAGING_ENABLE) {
                list.add(new RealPrinterServer(context, executorService, printer));
```

```java
public class RealPrinterServer extends BaseServer {
    private static final String TAG = "RealPrinterServer";
    private final UUID UUID_PRINTER = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private BluetoothAdapter mBluetoothAdapter;
    private BluetoothServerSocket mServerSocket;

    public RealPrinterServer(Context context, ExecutorService executorService, Printer printer) {
        this.mContext = context;
        this.mExecutorService = executorService;
        this.mPrinter = printer;
    }

    @Override // com.android.bluetooth.simulate.server.Server
    public void start() {
        this.mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        try {
            if (this.mServerSocket == null) {
                Log.v(TAG, "-Create ServerSocket-");
                this.mServerSocket = this.mBluetoothAdapter.listenUsingInsecureRfcommWithServiceRecord("Serveer", this.UUID_PRINTER);
                this.mExecutorService.execute(new BaseServer.ServerWork());
            } else {
                Log.v(TAG, "-Reuse ServerSocket-");
            }
```

# ~~Virtual~~ Bluetooth Printer

```
$ sdptools browse ▒▒:41:46:14:01:▒▒

  ...
  Service Name: Serveer
  Service RecHandle: 0x10005
  Service Class ID List:
    "Serial Port" (0x1101)
  Protocol Descriptor List:
    "L2CAP" (0x0100)
    "RFCOMM" (0x0003)
      Channel: 4
  Profile Descriptor List:
    "Serial Port" (0x1101)
      Version: 0x0102
  ...
```

# Demo

# Conclusion

- Crypto POS Companies should reconsider their device selection
  - A proven track record in security development
- Achieving security through obscurity may not be wise
  - Regular penetration testing and auditing should be the cornerstone
- A playful hacking platform for security researchers
  - Outdated Kernel, Bluetooth, WIFI, NFC, Baseband, Webview

# Questions?

@hhj4ck