Stealthily Access Your Android Phones: Bypass the Bluetooth Authentication

BlueRepli
by Sourcell Xu and Xin Xin
Who we are

Sourcell Xu

- IoT security researcher
- fO-000/bluescan
- Discovered of the BlueRepli
- sourcell.xu@dbappsecurity.com.cn

Xin Xin

- Hardware hacker
- Make the BlueRepli a convenient hardware tool
- xin.xin@dbappsecurity.com.cn
Chaotic Scenes of Privacy

• Access phone files 25,000 times in 10 minutes.

• Self-starting 7000 times and read the phone book in one hour.

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!
Could it be **Worse**?

No malicious or rogue apps installed

No touch

Android system

Wi-Fi
Mobile Network
NFC

Bluetooth

PBAP (Phone Book Access Profile)

MAP (Message Access Profile)
What’s Bluetooth Profile?

Bluetooth Profiles:
- L2CAP
- Link Manager
- Baseband
- Radio (PHY)

Bluetooth Protocol Layers:
- GAP
- Profile #1
- Profile #2
- Profile #3

Connections:
- PSE (Phonebook Server Equipment) via PBAP
- PCE (Phonebook Client Equipment) via PBAP
- MCE (Message Client Equipment) via MAP
- MSE (Message Server Equipment) via MAP
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 manufacture (may be affected 100 million devices)
- Totally Stealthily

contacts, call logs, short messages

attack

deceived

contacts, call logs, short messages

attack

caller

deceived

attack

fake short message

victim 1

or

victim 2
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?

- **Connect**
  - Secure Simple Pairing
  - Shared
    - New link key
      - Future authentication
      - Traffic encryption
  - No valid link key
  - New link key
The default IO capabilities of AOSP is DisplayYesNo
Bypass the **Pairing Request Dialog Box**
Side Effect of the Just Works Model

```c
/* if just_works and bonding bit is not set treat this as temporary */
if (p_ssp_cfm_req->just_works &&
    !(p_ssp_cfm_req->loc_auth_req & BTM_AUTH_BONDS) &&
    !(p_ssp_cfm_req->rmt_auth_req & BTM_AUTH_BONDS) &&
    !(check_cod((RawAddress*)p_ssp_cfm_req->bd_addr, COD_HID_POINTING)))
    pairing_cb.bond_type = BOND_TYPE_TEMPORARY;
else
    pairing_cb.bond_type = BOND_TYPE_PERSISTENT;
```
Why does the **Profile Access Request** pop up?

```
<xml version='1.0' encoding='utf-8' standalone='yes' >
<map />
</xml>
```

```
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;
}
```
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=":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;
    } else {
        return pref.getBoolean(device.getAddress(), false) ? BluetoothDevice.ACCESS_ALLOWED : BluetoothDevice.ACCESS_REJECTED;
    }
}
Bypass the **Profile Access Request** Dialog Box

```java
int getPhonebookAccessPermission(BluetoothDevice device) {
    enforceCallingOrSelfPermission(BLUETOOTH_PERM, "Need BLUETOOTH permission");
    SharedPreferences pref = getSharedPreferences(PHONEBOOK_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;
}
```

Address dependent?

PBAP

```bash
Manufacturer: Cambridge Silicon Radio (10)
Device address: 11:11:11:11:11:11
```

Address changed

MAP
Side Effect of the Just Works Model

```java
if (newState == BluetoothDevice.BOND_NONE) {
    mAdapterService.setPhonebookAccessPermission(dev, BTDevice.ACCESS_UNKNOWN);
    mAdapterService.setMessageAccessPermission(dev, BTDevice.ACCESS_UNKNOWN);
    mAdapterService.setSimAccessPermission(dev, BTDevice.ACCESS_UNKNOWN);
    // Set the profile Priorities to undefined
clearProfilePriority(dev);
}
else {
    BTIF_TRACE_DEBUG("Ns: Temporary key. Not storing, key_type=0x%04X, bond_type=0x%08x", _func_, __func__, p_auth_cmpl->key_type, pairing_cb.bond_type);
    if (pairing_cb.bond_type != BOND_TYPE_TEMPORARY) {
        BTIF_TRACE_DEBUG("Ns: sending BT_BOND_STATE_NONE for Temp pairing", _func_);
        btif_storage_remove_bonded_device(bd_addr);
        bond_state_changed(BT_STATUS_SUCCESS, bd_addr, BT_BOND_STATE_NONE);
    }
}
```

**No Input No Output** → **Just Works**

PBAP and MAP access permission cleared → **Temporary Bond**
Forge **CoD** to prevent passing **BT_BOND_STATE_NONE**

```c
RawAddress bd_addr = p_auth_cmpl->bd_addr;
if ((p_auth_cmpl->success) && (p_auth_cmpl->key_present)) {
    if ((p_auth_cmpl->key_type == HCI_KEY_TYPE_DEBUG_COMBO) ||
        (p_auth_cmpl->key_type == HCI_KEY_TYPE_AUTH_COMBO) ||
        (p_auth_cmpl->key_type == HCI_KEY_TYPE_CHANGED_COMBO) ||
        (p_auth_cmpl->key_type == HCI_KEY_TYPE_AUTH_COMBO_P_256)) {
        pairing_cb.bond_type = BT_BOND_STATE_NONE;
    }
    BTIF_TRACE_DEBUG("%s: Storing link key, key_type=0x%"X, bond_type=0x%"d",
                       _func__, p_auth_cmpl->key_type, pairing_cb.bond_type);
    ret = btif_storage_add_bonded_device(&bd_addr, p_auth_cmpl->key,
                       p_auth_cmpl->key_type, _func__, pairing_cb.bond_type,
                       bd_addr->pin_code_len);
    ASSERTC(ret == BT_STATUS_SUCCESS, "storing link key failed", ret);
} else {
    BTIF_TRACE_DEBUG("%s: Temporary key, Not storing, key_type=0x%"X, bond_type=0x%"d",
                       _func__, p_auth_cmpl->key_type, pairing_cb.bond_type);
    if (pairing_cb.bond_type == BOND_TYPE_TEMPORARY) {
        BTIF_TRACE_DEBUG("%s: Storing BT_BOND_STATE_NONE to Tmp pairing",
                       _func__);  
        btif_storage_remove_bonded_device(&bd_addr);
        bond_state_changed(BT_STATUS_SUCCESS, bd_addr, BT_BOND_STATE_NONE);
    }
    return;
}
/* if just works and bonding bit is not set treat this as temporary */
if (p_ssp_cfm_req->just_works &&
    (p_ssp_cfm_req->loc_auth_req & BTM_AUTH_BONDS) &&
    (p_ssp_cfm_req->rem_auth_req & BTM_AUTH_BONDS) &&
    (check_cfd((RawAddress*)&p_ssp_cfm_req->bd_addr, COD_HID_POINTING))) {
    pairing_cb.bond_type = BOND_TYPE_TEMPORARY;
} else {
    pairing_cb.bond_type = BOND_TYPE_PERSISTENT;

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

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

/* If JustWorks auto-accept */
if (p_ssp_cfm_req->just_works) {
    /* Pairing consent for JustWorks NOT needed if */
    /* 1. Incoming temporary pairing is detected */
    if (is_incoming && pairing_cb.bond_type == BOND_TYPE_TEMPORARY) {
        BTIF_TRACE_EVENT(
            "%s: Auto-accept JustWorks pairing for temporary incoming", __func__);
        btif_dm_ssp_reply(&bd_addr, BT_SSP_VARIANT_CONSENT, true, 0);
        return;
    }
}
```
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 ( ̄( ̄皿 ̄) ̄)↗

No BT_BOND_STATE_NONE
No BOND_NONE
No Permission clear

user number

About 65,900,000 results (0.53 seconds)

100 million users

officially announced that

Mar 23, 2020
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

  - contacts, call logs, short messages
  - fake short message

  or

  attack

  deceived

  attack

  contacts, call logs, short messages

  attack
Command Line Tool

Hardware Tool
Should we based on RaspberryPi?

- No battery support.
- Low integration, jumper wire everywhere.
- HDMI is not a good idea for portable device.
- SPI is too slow for higher resolution LCD panel.
- We just want a challenge.

Linux  Python  Bluetooth
Choose the solution.
Porting the bootloader an OS

Buildroot & U-Boot & Linux
Coding the GUI Interface.
Making a 3D printed shell.
The video demo
More security issues in the Bluetooth Profiles

<table>
<thead>
<tr>
<th>A2DP</th>
<th>CIP</th>
<th>FTP</th>
<th>HCRP</th>
<th>ICP</th>
<th>OPP</th>
<th>SDAP</th>
<th>WAPB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATT</td>
<td>CTP</td>
<td>GAVDP</td>
<td>HDP</td>
<td>LAP</td>
<td>PAN</td>
<td>SAP</td>
<td>UDI</td>
</tr>
<tr>
<td>AVRC</td>
<td>DIP</td>
<td>GAP</td>
<td>HFP</td>
<td>MESH</td>
<td>PBAP</td>
<td>SYNCH</td>
<td>ESDP</td>
</tr>
<tr>
<td>BIP</td>
<td>DUN</td>
<td>GATT</td>
<td>HID</td>
<td>MAP</td>
<td>PXP</td>
<td>SyncML</td>
<td>VCP</td>
</tr>
<tr>
<td>BPP</td>
<td>FAX</td>
<td>GOEP</td>
<td>HSP</td>
<td>OBEX</td>
<td>SPP</td>
<td>VDP</td>
<td>TAP</td>
</tr>
</tbody>
</table>
Thank you!

Any questions?

sourcell.xu@dbappsecurity.com.cn  xin.xin@dbappsecurity.com.cn