[IOT-594] Android Created: 16/Apr/21 Updated: 20/Aug/21

Status:	Assigned
Project:	IoT
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Task	**Priority:**	P3
Reporter:	Ruian Duan	**Assignee:**	Dan Regalado (Inactive)
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Attachments:	HFP_V1.7.0.pdf Screen Shot 2021-04-21 at 10.46.11 PM.png screenshot-1.png
Epic Link:	IoT Vulnerability Research
Product documentation or release notes:	No

Description

Bluetooth architecture: *https://source.android.com/devices/bluetooth*

The android source code scan results. I have cloned android source code on zmas server at **/home/rduan/android/source**.

For bluetooth scanning, the base path is **/home/rduan/android/source/system/bt** on zmas server.
For nfc scanning, the base path is **/home/rduan/android/source/system/nfc** on zmas server.

github link: https://github.com/lingfennan/android-scan
bluetooth scanning results: https://github.com/lingfennan/android-scan/blob/master/bt-cpp-security-results.csv
nfc scanning results: https://github.com/lingfennan/android-scan/blob/master/nfc-cpp-security-results.csv

The path works like this: for example, the first line in the results refers to path: **/stack/sdp/sdp_utils.cc**, the filepath on zmas server is then the base path + the reported path: **/home/rduan/android/source/system/bt/stack/sdp/sdp_utils.cc**

Comments

Comment by Ruian Duan [16/Apr/21]

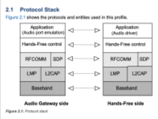
Two todo items:

- scan the whole android project for cpp code.
- scan the whole android project for java code.

Comment by Dan Regalado (Inactive) [22/Apr/21]

Related to Hands Free Profile:

Hands-Free control is the entity responsible for Hands-Free unit specific control signaling; this signaling is AT command based.



2.1 Protocol Stack

Figure 2.1 shows the protocols and entities used in this profile.

Figure 2.1: Protocol stack

The following roles are defined for this profile:

- Audio Gateway (AG) – This is the device that is the gateway of the audio, both for input and
  output. Typical devices acting as Audio Gateways are cellular phones.
- Hands-Free unit (HF) – This is the device acting as the Audio Gateway's remote audio input
  and output mechanism. It also provides some remote control means.

AT+BIEV (Bluetooth HF Indicators Feature)'

Syntax:
AT+BIEV= <assigned number>,<value> (Update value of indicator)

Description:
This command enables the HF to send updated values of the enabled HF indicators to the AG.
The AT+BIEV command shall not be used unless both AG and HF BRSF bits for the 'HF Indicators' feature are set to one.

Values:
<assigned number>: 0-65535, entered as a decimal unsigned integer without leading zeros, referencing an HF indicator assigned number.
Values are defined on the Bluetooth SIG Assigned Numbers [10] web page.

<value> 0 to 4,294,967,295, entered as a decimal unsigned integer without leading zeros. The meaning of the value depends of the <assigned number> and is defined on
the Bluetooth SIG Assigned Numbers [10] web page.

1 Enhanced Safety
0: Enhanced Safety is Disabled
1: Enhanced Safety is Enabled

2 Battery Level 0 ~ 100 : Remaining level of Battery

4.2.1.4 HF Indicators

- If the HF supports the HF indicator feature, it shall check the +BRSF response to see if the AG
  also supports the HF Indicator feature.
- If both the HF and AG support the HF Indicator feature, then the HF shall send the
  AT+BIND=<HF supported HF indicators> command to the AG to notify the AG of the supported
  indicators' assigned numbers in the HF. The AG shall respond with OK.
  2 Legacy devices shall not indicate support for the Codec Negotiation Feature
  Bluetooth SIG Proprietary Page 24 of 144
- After having provided the AG with the HF indicators it supports, the HF shall send the
  AT+BIND=? to request HF indicators supported by the AG. The AG shall reply with the +BIND
  response listing all HF indicators that it supports followed by an OK.
- Once the HF receives the supported HF indicators list from the AG, the HF shall send the
  AT+BIND? command to determine which HF indicators are enabled. The AG shall respond with
  one or more +BIND responses. The AG shall terminate the list with OK. (See Section 4.35.1.5).
- From this point onwards, the HF may send the **AT+BIEV** command with the corresponding HF indicator value whenever a change in value occurs of an enabled HF indicator.
- The AG may enable or disable the notification of any HF indicator at any time by using the
  +BIND unsolicited response (See Section 4.35.1.4).

- BUG ***************
  "Potentially overrunning write","Buffer write operations that do not control the length of data written may overflow.","error","This 'call to sprintf' operation requires 33 bytes but the destination is only 32 bytes.","/bta/hf_client/bta_hf_client_at.cc","1864","13","1864","19"

Those two values are integer ones so,

```
  int len = sprintf(buf, "AT+BIEV=%d,%d\r", indicator_id, indicator_value);
```

Since the request is sent by HF client, it should crash itself in the worst case escenario, since it sets those values.
No bug identified.

Comment by Dan Regalado (Inactive) [22/Apr/21]

"Call to memory access function may overflow buffer","Incorrect use of a function that accesses a memory buffer may read or write data past the end of that buffer.","recommendation","This 'memcmp' operation may access 16 bytes but the [[""second buffer""|""relative:///stack/include/sdp_api.h:86:13:86:17""]] is only 4 bytes.","/stack/sdp/sdp_utils.cc","949","7","949","12"

```
bool sdpu_compare_uuid_with_attr(const Uuid& uuid, tSDP_DISC_ATTR* p_attr) {
  int len = uuid.GetShortestRepresentationSize();
  if (len == 2) return uuid.As16Bit() == p_attr->attr_value.v.u16;
  if (len == 4) return uuid.As32Bit() == p_attr->attr_value.v.u32;
  if (memcmp(uuid.To128BitBE().data(), (void*)p_attr->attr_value.v.array,
             Uuid::kNumBytes128) == 0)
    return (true);

  return (false);
```

Do not see too much value to eventually return always false if can trigger it.

Comment by Dan Regalado (Inactive) [22/Apr/21]

"Potential integer arithmetic overflow","A user-controlled integer arithmetic expression that is not validated can cause overflows.","warning","[[""User-provided value""|""relative:///osi/src/osi.cc:45:38:45:42""]] flows to here and is used in an expression which might overflow.","/osi/src/allocation_tracker.cc","57","62","57","69"

```
int osi_rand(void) {
  int rand;
  int rand_fd = open(RANDOM_PATH, O_RDONLY);
  if (rand_fd == INVALID_FD) {
    LOG_ERROR("%s can't open rand fd %s: %s ", __func__, RANDOM_PATH,
              strerror(errno));
    CHECK(rand_fd != INVALID_FD);
  }

  ssize_t read_bytes = read(rand_fd, &rand, sizeof(rand));     <-------------------HERE - NOT CONTROL BY USER
  close(rand_fd);

  CHECK(read_bytes == sizeof(rand));

  if (rand < 0) rand = -rand;

  return rand;
}
```

Comment by Dan Regalado (Inactive) [22/Apr/21]

Nothing else interested seen in the bluetooth and nfc hits

Comment by Vijay Prakash [30/Apr/21]

```
Line 18:
"Potential integer arithmetic overflow","A user-controlled integer arithmetic expression that is not validated can cause overflows.","warning","[[""User-provided value""|""relative:///osi/src/socket.cc:135:38:135:40""]] flows to here and is used
in an expression which might overflow.","/hci/src/hci_inject.cc","163","3","163","28"
```

Definition of the structure in question:

```
 45 typedef struct {
 46   socket_t* socket;
 47   uint8_t buffer[65536 + 3];  // 2 bytes length prefix, 1 byte type prefix.
 48   size_t buffer_size;
 49 } client_t;
```

Code lines surrounding the line where the issue is reported

```
150 static void read_ready(UNUSED_ATTR socket_t* socket, void* context) {
151   CHECK(socket != NULL);
152   CHECK(context != NULL);
153
154   client_t* client = (client_t*)context;
155
156   ssize_t ret =
157       socket_read(client->socket, client->buffer + client->buffer_size,
158                   sizeof(client->buffer) - client->buffer_size); <--------------passed size to read is restricted by the buffer size
159   if (ret == 0 || (ret == -1 && ret != EWOULDBLOCK && ret != EAGAIN)) {
160     list_remove(clients, client);
161     return;
162   }
163   client->buffer_size += ret; <---------------------reported issue
```

Function that works on variable the reported in the issue

```
130 ssize_t socket_read(const socket_t* socket, void* buf, size_t count) {
131   CHECK(socket != NULL);
132   CHECK(buf != NULL);
133
134   ssize_t ret;
135   OSI_NO_INTR(ret = recv(socket->fd, buf, count, MSG_DONTWAIT)); <----------- can't read more than size of buffer
136
137   return ret;
138 }
```

As recv function can't read more than 65536, which could fit in client->buffer_size without any issue because the type of buffer_size is size_t. So, this reported issue will not result in an overflow.

Comment by Vijay Prakash [30/Apr/21]

```
Line:19
"Potential integer arithmetic overflow","A user-controlled integer arithmetic expression that is not validated can cause overflows.","warning","[[""User-provided value""|""relative:///osi/src/socket.cc:135:38:135:40""]] flows to here and is used
in an expression which might overflow negatively.","/hci/src/hci_inject.cc","192","5","192","36"
```

Code lines surrounding the reported issue

```
165   while (client->buffer_size > 3) {
166     uint8_t* buffer = client->buffer;
167     hci_packet_t packet_type = (hci_packet_t)buffer[0];
168     size_t packet_len = (buffer[2] << 8) | buffer[1];
169     size_t frame_len = 3 + packet_len;
170
171     if (client->buffer_size < frame_len) break;
172
173     // TODO(sharvil): validate incoming HCI messages.
174     // TODO(sharvil): once we have an HCI parser, we can eliminate
175     //   the 2-byte size field since it will be contained in the packet.
176
177     BT_HDR* buf = (BT_HDR*)buffer_allocator->alloc(BT_HDR_SIZE + packet_len);
178     if (buf) {
179       buf->event = hci_packet_to_event(packet_type);
180       buf->offset = 0;
181       buf->layer_specific = 0;
182       buf->len = packet_len;
183       memcpy(buf->data, buffer + 3, packet_len);
184       hci->transmit_downward(buf->event, buf);
185     } else {
186       LOG_ERROR("%s dropping injected packet of length %zu", __func__,
187                 packet_len);
```

At line 169 frame_len is set to 3 + packet_len. packet_len is read from the packet buffer, which is controlled by user data sent as a packet.

packet_len variable is of type size_t and its max value could be of unsigned int of 2 bytes, which would definitely fit in size_t at any platform.

size_t is unsigned int.

If size_t is of 2 bytes frame_len could overflow at line 169. This could result in an issue where int is 2 bytes, whereas if it's 4 bytes there will not be any problem.

- 4-byte int scenario:*

As packe_ len is not verified by the read length at line 168, packet_len could be the maximum value stored in 2 bytes and frame_len would be 3 + packet_len. If a packet contains a small amount of buffer data or anything smaller than packet_len the check at line 171 will break the while loop. In the scenario when buffer_size is more than frame_len, the case purposefully packet_len is sent small but data in the packet is larger than packt_len, this case will not cause an issue as well because the only packet_len of data will be stored in the buffer and the rest of the data will be treated as another packet. Considering both the cases no chances of security issues here.

- 2-byte int scenario:*
  Although unlikely, integer size could be 2 bytes. In this case, if packet_len is the maximum of 2 bytes then frame_len will overflow but data in the packet will be parsed correctly. There would be an issue between 190-192 because frame_len is smaller after the overflow and will result in re-parsing the packet, leading to sending an unwanted already parsed packet at transmit_downward at line 184. Although, this issue doesn't seem to interest me because of the platform dependency.

Another scenario when packet_len is 0:
In this case, frame_len will be 3. If buffer->size is more than 3 control flow will go after line 171. And between 190-192 buffer data and buffer->size will be updated. If the buffer contains more than 3 bytes of data, while loop will continue until the buffer contains less than 3 bytes of data. So, no chances of an infinite loop in this scenario.

Comment by Vijay Prakash [04/May/21]

Issue:

```
"Potential integer arithmetic overflow","A user-controlled integer arithmetic expression that is not validated can cause overflows.","warning","[[""User-provided value""|""relative:///btif/co/bta_hh_co.cc:87:37:87:39""]] flows to here and is
used in an expression which might overflow.","/osi/src/allocation_tracker.cc","173","30","173","53"
```

Code nearby the reported issue:

```
172 size_t allocation_tracker_resize_for_canary(size_t size) {
173   return (!enabled) ? size : size + (2 * canary_size);
174 }


In same file at line 40:
 40 static const size_t canary_size = 8;
```

Malloc and calloc in file **./osi/src/allocator.cc** calls the function reported in the issue:

```
 58 void* osi_malloc(size_t size) {
 59   size_t real_size = allocation_tracker_resize_for_canary(size);
 60   void* ptr = malloc(real_size);
 61   CHECK(ptr);
 62   return allocation_tracker_notify_alloc(alloc_allocator_id, ptr, size);
 63 }


void* osi_calloc(size_t size) {
  size_t real_size = allocation_tracker_resize_for_canary(size);
  void* ptr = calloc(1, real_size);
  CHECK(ptr);
  return allocation_tracker_notify_alloc(alloc_allocator_id, ptr, size);
}
```

In file allocation_tracker_notify_alloc function in `./osi/src/allocation_tracker.cc`:

```
 99 void* allocation_tracker_notify_alloc(uint8_t allocator_id, void* ptr,
100                                        size_t requested_size) {
```

```
101   char* return_ptr;
102   {
103     std::unique_lock<std::mutex> lock(tracker_lock);
104     if (!enabled || !ptr) return ptr;
105
106     // Keep statistics
107     alloc_counter++;
108     alloc_total_size += allocation_tracker_resize_for_canary(requested_size);
109
110     return_ptr = ((char*)ptr) + canary_size;
111
112     auto map_entry = allocations.find(return_ptr);
113     allocation_t* allocation;
114     if (map_entry != allocations.end()) {
115       allocation = map_entry->second;
116       CHECK(allocation->freed);  // Must have been freed before
117     } else {
118       allocation = (allocation_t*)calloc(1, sizeof(allocation_t));
119       allocations[return_ptr] = allocation;
120     }
```

Code line which starts the control flow in this issue:

```
154         uint32_t* get_rpt_id = (uint32_t*)osi_malloc(sizeof(uint32_t));
```

As this malloc asks 4 bytes for allocation, even after adding 8 bytes there will be no overflow.

TODO: look for all the malloc and calloc in the project where user input data is used in malloc or calloc. This could potentially lead to integer overflow.

Comment by Vijay Prakash [11/May/21]

One interesting use of osi_malloc: `./btif/src/btif_sdp_server.cc:196: bluetooth_sdp_record* record = (bluetooth_sdp_record*)osi_malloc(record_size);`
command used to find it: `grep -rn osi_malloc --include=`**`.cc --include=`**`.h .`

Surrounding code:

```
185 /* Reserve a slot in sdp_slots, copy data and set a reference to the copy.
186  * The record_data will contain both the record and any data pointed to by
187  * the record.
188  * Currently this covers:
189  *   service_name string,
190  *   user1_ptr and
191  *   user2_ptr. */
192 static int alloc_sdp_slot(bluetooth_sdp_record* in_record) {
193   int record_size = get_sdp_records_size(in_record, 1); <------------- this is of interest
194   /* We are optimists here, and preallocate the record.
195    * This is to reduce the time we hold the sdp_lock. */
196   bluetooth_sdp_record* record = (bluetooth_sdp_record*)osi_malloc(record_size);
197
198   copy_sdp_records(in_record, record, 1);
199   {
200     std::unique_lock<std::recursive_mutex> lock(sdp_lock);
201     for (int i = 0; i < MAX_SDP_SLOTS; i++) {
202       if (sdp_slots[i].state == SDP_RECORD_FREE) {
203         sdp_slots[i].state = SDP_RECORD_ALLOCED;
204         sdp_slots[i].record_data = record;
205         return i;
206       }
```

Definition of the function `get_sdp_records_size` in the same file:

```
123 int get_sdp_records_size(bluetooth_sdp_record* in_record, int count) {
124   bluetooth_sdp_record* record = in_record; <-------- structure of interest
125   int records_size = 0;
126   int i;
127   for (i = 0; i < count; i++) {
128     record = &in_record[i];
129     records_size += sizeof(bluetooth_sdp_record);
130     records_size += record->hdr.service_name_length;
131     if (record->hdr.service_name_length > 0) {
132       records_size++; /* + '\0' termination of string */
133     }
134     records_size += record->hdr.user1_ptr_len;
135     records_size += record->hdr.user2_ptr_len;
136   }
137   return records_size;
138 }
```

Declaration of bluetooth_sdp_record in file ./include/hardware/bt_sdp.h:

```
111 typedef union {
112   bluetooth_sdp_hdr_overlay hdr; <------ field of interest
113   bluetooth_sdp_mas_record mas;
114   bluetooth_sdp_mns_record mns;
115   bluetooth_sdp_pse_record pse;
116   bluetooth_sdp_pce_record pce;
117   bluetooth_sdp_ops_record ops;
118   bluetooth_sdp_sap_record sap;
119   bluetooth_sdp_dip_record dip;
120 } bluetooth_sdp_record;
```

Declaration of bluetooth_sdp_hdr_overlay in the same file:

```
 49 /**
 50  * Some signals need additional pointers, hence we introduce a
 51  * generic way to handle these pointers.
 52  */
 53 typedef struct _bluetooth_sdp_hdr_overlay {
 54   bluetooth_sdp_types type;
 55   bluetooth::Uuid uuid;
 56   uint32_t service_name_length; <--------- used in calculating records_size in function get_sdp_records_size
 57   char* service_name;
 58   int32_t rfcomm_channel_number;
 59   int32_t l2cap_psm;
 60   int32_t profile_version;
 61
 62   // User pointers, only used for some signals - see bluetooth_sdp_ops_record
 63   int user1_ptr_len; <--------- used in calculating records_size in function get_sdp_records_size
 64   uint8_t* user1_ptr;
 65   int user2_ptr_len;<--------- used in calculating records_size in function get_sdp_records_size
 66   uint8_t* user2_ptr;
 67 } bluetooth_sdp_hdr_overlay;
```

Comment by Vijay Prakash [13/May/21]

Here is an example of creating a discoverable service https://stackoverflow.com/questions/30813854/how-do-bluetooth-sdp-and-uuids-work-specifically-for-android.

Comment by Vijay Prakash [17/May/21]

git clone https://android.googlesource.com/platform/packages/apps/Bluetooth (Bluetooth Process)

An e.g. of JNI in Bluetooth process communicating with Bluetooth stack
jni/com_android_bluetooth_sdp.cpp

```
444 static jint sdpCreateSapsRecordNative(JNIEnv* env, jobject obj,
445                                       jstring name_str, jint scn,
446                                       jint version) {
447   ALOGD("%s", __func__);
448   if (!sBluetoothSdpInterface) return -1;
449
450   bluetooth_sdp_record record = {};  // Must be zero initialized
451   record.sap.hdr.type = SDP_TYPE_SAP_SERVER;
452
453   const char* service_name = NULL;
454   if (name_str != NULL) {
455     service_name = env->GetStringUTFChars(name_str, NULL);
456     record.mas.hdr.service_name = (char*)service_name;
457     record.mas.hdr.service_name_length = strlen(service_name); <-------------length check
458   } else {
459     record.mas.hdr.service_name = NULL;
460     record.mas.hdr.service_name_length = 0;
461   }
462   record.mas.hdr.rfcomm_channel_number = scn;
463   record.mas.hdr.profile_version = version;
464
465   int handle = -1;
```

Binding of native methods with Java Bluetooth process

```
513 static JNINativeMethod sMethods[] = {
514     /* name, signature, funcPtr */
515     {"classInitNative", "()V", (void*)classInitNative},
516     {"initializeNative", "()V", (void*)initializeNative},
517     {"cleanupNative", "()V", (void*)cleanupNative},
518     {"sdpSearchNative", "([B[B)Z", (void*)sdpSearchNative},
519     {"sdpCreateMapMasRecordNative", "(Ljava/lang/String;IIIIII)I",
520      (void*)sdpCreateMapMasRecordNative},
521     {"sdpCreateMapMnsRecordNative", "(Ljava/lang/String;IIII)I",
522      (void*)sdpCreateMapMnsRecordNative},
    {"sdpCreatePbapPceRecordNative", "(Ljava/lang/String;I)I",
     (void*)sdpCreatePbapPceRecordNative},
    {"sdpCreatePbapPseRecordNative", "(Ljava/lang/String;IIIII)I",
     (void*)sdpCreatePbapPseRecordNative},
    {"sdpCreateOppOpsRecordNative", "(Ljava/lang/String;III[B)I",
     (void*)sdpCreateOppOpsRecordNative},
    {"sdpCreateSapsRecordNative", "(Ljava/lang/String;II)I",
     (void*)sdpCreateSapsRecordNative},
    {"sdpRemoveSdpRecordNative", "(I)Z", (void*)sdpRemoveSdpRecordNative}};

int register_com_android_bluetooth_sdp(JNIEnv* env) {
  return jniRegisterNativeMethods(env, "com/android/bluetooth/sdp/SdpManager",
```

Bluetooth process calling native api
./src/com/android/bluetooth/sdp/SdpManager.java

```
    public int createSapsRecord(String serviceName, int rfcommChannel, int version) {
        if (!sNativeAvailable) {
            throw new RuntimeException(TAG + " sNativeAvailable == false - native not initializ    ed");
        }
        return sdpCreateSapsRecordNative(serviceName, rfcommChannel, version); <----------native call
    }
```

SAP service implementation in Bluetooth process.
It communicates with the Application framework with binder mechanism

./src/com/android/bluetooth/sap/SapService.java

```
public class SapService extends ProfileService {

    private static final String SDP_SAP_SERVICE_NAME = "SIM Access";
    private static final int SDP_SAP_VERSION = 0x0102;
```

```
    private static final int CREATE_RETRY_TIME = 10;

    private boolean initSocket() {
        if (VERBOSE) {
            Log.v(TAG, "Sap Service initSocket");
        }

        boolean initSocketOK = false;

        // It's possible that create will fail in some cases. retry for 10 times
        for (int i = 0; i < CREATE_RETRY_TIME && !mInterrupted; i++) {
            initSocketOK = true;
            try {
                // It is mandatory for MSE to support initiation of bonding and encryption.
                // TODO: Consider reusing the mServerSocket - it is indented to be reused
                //       for multiple connections.
                mServerSocket = mAdapter.listenUsingRfcommOn(
                        BluetoothAdapter.SOCKET_CHANNEL_AUTO_STATIC_NO_SDP, true, true);
                removeSdpRecord();
                mSdpHandle = SdpManager.getDefaultManager()
```

Bluetoooth service implementation is at src/com/android/bluetooth/btservice/AdapterService.java

Android Bluetooth framework is located at:
git clone https://android.googlesource.com/platform/frameworks/base

./services/core/java/com/android/server/BluetoothManagerService.java (service)

Bluetooth service communication: ./services/core/java/com/android/server/BluetoothService.java

```
class BluetoothService extends SystemService {
    private BluetoothManagerService mBluetoothManagerService;
    private boolean mInitialized = false;

    public BluetoothService(Context context) {
        super(context);
        mBluetoothManagerService = new BluetoothManagerService(context);
    }

    private void initialize() {
        if (!mInitialized) {
            mBluetoothManagerService.handleOnBootPhase();
            mInitialized = true;
        }
    }

    @Override
    public void onStart() {
    }

    @Override
    public void onBootPhase(int phase) {
```

./services/core/java/com/android/server/BluetoothManagerService.java is the file in the application framework
that manages the service listens to requests from applications and communicates with Bluetooth services with the binder mechanism.

Because of the length calculation in JNI we can't pass different service name lengths from their actual length because but we still need to check if we pass overly long service length, would it lead to a crash?

Comment by Vijay Prakash [17/May/21]

git clone https://android.googlesource.com/platform/packages/apps/Bluetooth (Bluetooth Process)

An e.g. of JNI in Bluetooth process communicating with Bluetooth stack
jni/com_android_bluetooth_sdp.cpp

```
static jint sdpCreateSapsRecordNative(JNIEnv* env, jobject obj,
                                      jstring name_str, jint scn,
                                      jint version) {
  ALOGD("%s", __func__);
  if (!sBluetoothSdpInterface) return -1;

  bluetooth_sdp_record record = {};  // Must be zero initialized
  record.sap.hdr.type = SDP_TYPE_SAP_SERVER;

  const char* service_name = NULL;
  if (name_str != NULL) {
    service_name = env->GetStringUTFChars(name_str, NULL);
    record.mas.hdr.service_name = (char*)service_name;
    record.mas.hdr.service_name_length = strlen(service_name); <-------------length check
  } else {
    record.mas.hdr.service_name = NULL;
    record.mas.hdr.service_name_length = 0;
  }
  record.mas.hdr.rfcomm_channel_number = scn;
  record.mas.hdr.profile_version = version;

  int handle = -1;
```

Binding of native methods with Java Bluetooth process

```
static JNINativeMethod sMethods[] = {
    /* name, signature, funcPtr */
    {"classInitNative", "()V", (void*)classInitNative},
    {"initializeNative", "()V", (void*)initializeNative},
    {"cleanupNative", "()V", (void*)cleanupNative},
    {"sdpSearchNative", "([B[B)Z", (void*)sdpSearchNative},
    {"sdpCreateMapMasRecordNative", "(Ljava/lang/String;IIIIII)I",
     (void*)sdpCreateMapMasRecordNative},
    {"sdpCreateMapMnsRecordNative", "(Ljava/lang/String;IIII)I",
     (void*)sdpCreateMapMnsRecordNative},
    {"sdpCreatePbapPceRecordNative", "(Ljava/lang/String;I)I",
     (void*)sdpCreatePbapPceRecordNative},
    {"sdpCreatePbapPseRecordNative", "(Ljava/lang/String;IIIII)I",
     (void*)sdpCreatePbapPseRecordNative},
    {"sdpCreateOppOpsRecordNative", "(Ljava/lang/String;III[B)I",
     (void*)sdpCreateOppOpsRecordNative},
    {"sdpCreateSapsRecordNative", "(Ljava/lang/String;II)I",
     (void*)sdpCreateSapsRecordNative},
    {"sdpRemoveSdpRecordNative", "(I)Z", (void*)sdpRemoveSdpRecordNative}};

int register_com_android_bluetooth_sdp(JNIEnv* env) {
  return jniRegisterNativeMethods(env, "com/android/bluetooth/sdp/SdpManager",
```

Bluetooth process calling native api
./src/com/android/bluetooth/sdp/SdpManager.java

```
    public int createSapsRecord(String serviceName, int rfcommChannel, int version) {
        if (!sNativeAvailable) {
            throw new RuntimeException(TAG + " sNativeAvailable == false - native not initializ    ed");
        }
        return sdpCreateSapsRecordNative(serviceName, rfcommChannel, version); <----------native call
    }
```

SAP service implementation in Bluetooth process.
It communicates with the Application framework with binder mechanism

./src/com/android/bluetooth/sap/SapService.java

```

    private static final String SDP_SAP_SERVICE_NAME = "SIM Access";
    private static final int SDP_SAP_VERSION = 0x0102;


    private static final int CREATE_RETRY_TIME = 10;

    private boolean initSocket() {
        if (VERBOSE) {
            Log.v(TAG, "Sap Service initSocket");
        }

        boolean initSocketOK = false;

        // It's possible that create will fail in some cases. retry for 10 times
        for (int i = 0; i < CREATE_RETRY_TIME && !mInterrupted; i++) {
            initSocketOK = true;
            try {
                // It is mandatory for MSE to support initiation of bonding and encryption.
                // TODO: Consider reusing the mServerSocket - it is indented to be reused
                //       for multiple connections.
                mServerSocket = mAdapter.listenUsingRfcommOn(
                        BluetoothAdapter.SOCKET_CHANNEL_AUTO_STATIC_NO_SDP, true, true);
```

Android Bluetooth framework is located at:
git clone https://android.googlesource.com/platform/frameworks/base

./services/core/java/com/android/server/BluetoothManagerService.java (service)

Bluetooth service implementation: ./services/core/java/com/android/server/BluetoothService.java

```
class BluetoothService extends SystemService {
    private BluetoothManagerService mBluetoothManagerService;
    private boolean mInitialized = false;

    public BluetoothService(Context context) {
        super(context);
        mBluetoothManagerService = new BluetoothManagerService(context);
    }

    private void initialize() {
        if (!mInitialized) {
            mBluetoothManagerService.handleOnBootPhase();
            mInitialized = true;
        }
    }

    @Override
    public void onStart() {
    }

    @Override
    public void onBootPhase(int phase) {
        if (phase == SystemService.PHASE_SYSTEM_SERVICES_READY) {
```

in Bluetooth service:
./src/com/android/bluetooth/sap/SapService.java

```
    private static final String SDP_SAP_SERVICE_NAME = "SIM Access";
    private static final int SDP_SAP_VERSION = 0x0102;


    private static final int CREATE_RETRY_TIME = 10;

    private boolean initSocket() {
        if (VERBOSE) {
            Log.v(TAG, "Sap Service initSocket");
        }

        boolean initSocketOK = false;

        // It's possible that create will fail in some cases. retry for 10 times
        for (int i = 0; i < CREATE_RETRY_TIME && !mInterrupted; i++) {
            initSocketOK = true;
            try {
                // It is mandatory for MSE to support initiation of bonding and encryption.
                // TODO: Consider reusing the mServerSocket - it is indented to be reused
                //       for multiple connections.
                mServerSocket = mAdapter.listenUsingRfcommOn(   <--------------------- listen call in framework
                        BluetoothAdapter.SOCKET_CHANNEL_AUTO_STATIC_NO_SDP, true, true);
                removeSdpRecord();
```

https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/bluetooth/ (framework)

https://android.googlesource.com/platform/packages/apps/Bluetooth/+/refs/heads/master/src/com/android/bluetooth/sap/SapService.java

```
    @Override
    protected boolean start() {
        Log.v(TAG, "start()");
        IntentFilter filter = new IntentFilter();
        filter.addAction(BluetoothDevice.ACTION_CONNECTION_ACCESS_REPLY);
        filter.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
        filter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
        filter.addAction(USER_CONFIRM_TIMEOUT_ACTION);
        try {
            registerReceiver(mSapReceiver, filter);
            mIsRegistered = true;
        } catch (Exception e) {
            Log.w(TAG, "Unable to register sap receiver", e);
        }
        mInterrupted = false;
        mAdapter = BluetoothAdapter.getDefaultAdapter();   <----------------------- madapter is initialized
        // start RFCOMM listener
        mSessionStatusHandler.sendMessage(mSessionStatusHandler.obtainMessage(START_LISTENER));
        setSapService(this);
        return true;
    }
```

https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/bluetooth/BluetoothAdapter.java#2490

```
    public BluetoothServerSocket listenUsingRfcommOn(int channel, boolean mitm,
            boolean min16DigitPin) throws IOException {
        BluetoothServerSocket socket =
                new BluetoothServerSocket(BluetoothSocket.TYPE_RFCOMM, true, true, channel, mitm,
                        min16DigitPin);
        int errno = socket.mSocket.bindListen();
        if (channel == SOCKET_CHANNEL_AUTO_STATIC_NO_SDP) {
            socket.setChannel(socket.mSocket.getPort());
        }
        if (errno != 0) {
            //TODO(BT): Throw the same exception error code
            // that the previous code was using.
            //socket.mSocket.throwErrnoNative(errno);
            throw new IOException("Error: " + errno);
        }
        return socket;
    }
```

https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/bluetooth/BluetoothServerSocket.java#118

```
    /*package*/ BluetoothServerSocket(int type, boolean auth, boolean encrypt, int port,
            boolean mitm, boolean min16DigitPin)
            throws IOException {
        mChannel = port;
        mSocket = new BluetoothSocket(type, -1, auth, encrypt, null, port, null, mitm,
                min16DigitPin);
        if (port == BluetoothAdapter.SOCKET_CHANNEL_AUTO_STATIC_NO_SDP) {
            mSocket.setExcludeSdp(true);
        }
    }
```

https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/bluetooth/BluetoothSocket.java#202

```
    /*package*/ BluetoothSocket(int type, int fd, boolean auth, boolean encrypt,
            BluetoothDevice device, int port, ParcelUuid uuid, boolean mitm, boolean min16DigitPin)
            throws IOException {
        if (VDBG) Log.d(TAG, "Creating new BluetoothSocket of type: " + type);
        if (type == BluetoothSocket.TYPE_RFCOMM && uuid == null && fd == -1
                && port != BluetoothAdapter.SOCKET_CHANNEL_AUTO_STATIC_NO_SDP) {
            if (port < 1 || port > MAX_RFCOMM_CHANNEL) {
                throw new IOException("Invalid RFCOMM channel: " + port);
            }
        }
        if (uuid != null) {
            mUuid = uuid;
        } else {
            mUuid = new ParcelUuid(new UUID(0, 0));
        }
        mType = type;
        mAuth = auth;
        mAuthMitm = mitm;
        mMin16DigitPin = min16DigitPin;
        mEncrypt = encrypt;
        mDevice = device;
        mPort = port;
```

https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/bluetooth/BluetoothSocket.java#92

```
    public static final int MAX_RFCOMM_CHANNEL = 30;
    /*package*/ static final int MAX_L2CAP_PACKAGE_SIZE = 0xFFFF;
    /** RFCOMM socket */
    public static final int TYPE_RFCOMM = 1;  <---------------------------------value of RFCOMM channel
    /** SCO socket */
    public static final int TYPE_SCO = 2;
    /** L2CAP socket */
    public static final int TYPE_L2CAP = 3;
    /** L2CAP socket on BR/EDR transport
     * @hide
     */
    public static final int TYPE_L2CAP_BREDR = TYPE_L2CAP;
    /** L2CAP socket on LE transport
     * @hide
     */
    public static final int TYPE_L2CAP_LE = 4;
```

./services/core/java/com/android/server/BluetoothManagerService.java is the file in the application framework
that manages the service listens to requests from applications and communicates with Bluetooth services with the binder mechanism.

Because of the length calculation in JNI we can't pass different service name lengths from their actual length but we still need to check if we pass overly long service length, would it lead to crash?

Comment by Vijay Prakash [17/May/21]

In framework:
core/java/android/bluetooth/BluetoothAdapter.java needs to figure out how SDP record get added automatically for the UUID

```
    /**
     * Create a listening, secure RFCOMM Bluetooth socket with Service Record.
     * <p>A remote device connecting to this socket will be authenticated and
     * communication on this socket will be encrypted.
     * <p>Use {@link BluetoothServerSocket#accept} to retrieve incoming
     * connections from a listening {@link BluetoothServerSocket}.
     * <p>The system will assign an unused RFCOMM channel to listen on.
     * <p>The system will also register a Service Discovery
     * Protocol (SDP) record with the local SDP server containing the specified
     * UUID, service name, and auto-assigned channel. Remote Bluetooth devices
     * can use the same UUID to query our SDP server and discover which channel
     * to connect to. This SDP record will be removed when this socket is
     * closed, or if this application closes unexpectedly.
     * <p>Use {@link BluetoothDevice#createRfcommSocketToServiceRecord} to
     * connect to this socket from another device using the same {@link UUID}.
     *
     * @param name service name for SDP record
     * @param uuid uuid for SDP record
     * @return a listening RFCOMM BluetoothServerSocket
     * @throws IOException on error, for example Bluetooth not available, or insufficient
     * permissions, or channel in use.
     */
```

```
    private BluetoothServerSocket createNewRfcommSocketAndRecord(String name, UUID uuid,
            boolean auth, boolean encrypt) throws IOException {
        BluetoothServerSocket socket;
        socket = new BluetoothServerSocket(BluetoothSocket.TYPE_RFCOMM, auth, encrypt,
                new ParcelUuid(uuid));
        socket.setServiceName(name);
        int errno = socket.mSocket.bindListen();
        if (errno != 0) {
            //TODO(BT): Throw the same exception error code
            // that the previous code was using.
            //socket.mSocket.throwErrnoNative(errno);
            throw new IOException("Error: " + errno);
        }
        return socket;
    }
```

core/java/android/bluetooth/BluetoothServerSocket.java

```
    /**
     * Construct a socket for incoming connections.
     *
     * @param type type of socket
     * @param auth require the remote device to be authenticated
     * @param encrypt require the connection to be encrypted
     * @param uuid uuid
     * @throws IOException On error, for example Bluetooth not available, or insuffici    ent
     * privileges
     */
    /*package*/ BluetoothServerSocket(int type, boolean auth, boolean encrypt, ParcelU    uid uuid)
            throws IOException {
        mSocket = new BluetoothSocket(type, -1, auth, encrypt, null, -1, uuid);
        // TODO: This is the same as mChannel = -1 - is this intentional?
        mChannel = mSocket.getPort();
    }
```

./core/java/android/bluetooth/BluetoothSocket.java

```
    /**
     * Construct a BluetoothSocket.
     *
     * @param type type of socket
     * @param fd fd to use for connected socket, or -1 for a new socket
     * @param auth require the remote device to be authenticated
     * @param encrypt require the connection to be encrypted
     * @param device remote device that this socket can connect to
     * @param port remote port
     * @param uuid SDP uuid
     * @throws IOException On error, for example Bluetooth not available, or insufficient
     * privileges
     */
    /*package*/ BluetoothSocket(int type, int fd, boolean auth, boolean encrypt,
            BluetoothDevice device, int port, ParcelUuid uuid) throws IOException {
        this(type, fd, auth, encrypt, device, port, uuid, false, false);
    }
```

Comment by Vijay Prakash [01/Jun/21]

Created an android application that can act as Bluetooth server, and tried to create a server with the name >= 2^32 - 8 to trigger the vulnerability. Due to the large memory requirements app kept crashing. I will try to figure out a different way to set the name to check if it can trigger the crash in osi_malloc.

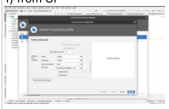
Comment by Ruian Duan [01/Jun/21]

Hi Vijay Prakash, the memory size limitation might be enforced by the android framework. One solution I can think of is to use NDK to write and compile a native binary to link against **libbluetooth.so**.

I haven't verified if the following code works or not though.
https://stackoverflow.com/questions/12552868/how-do-i-include-bluetooth-bluetooth-h-for-ndk-toolchains-gcc

Comment by Vijay Prakash [02/Jun/21]

Tried increasing the RAM and heap size
1) from UI



2) from command line/.ini file by specifying explicitly in `.android/avd/3.7_WVGA_Nexus_One_Edited_API_30.ini`

```
vm.heapSize=40960
hw.ramSize=41984
```

But still couldn't get more than 576 MB of heap size.

Going to proceed with NDK way.

Comment by Vijay Prakash [02/Jun/21]

NDK and CMake installation steps https://developer.android.com/studio/projects/install-ndk#groovy.

Comment by Vijay Prakash [03/Jun/21]

An example of how libbluetooth.so can be loaded https://android.googlesource.com/platform/packages/apps/Bluetooth/+/refs/heads/master/jni/com_android_bluetooth_btservice_AdapterService.cpp#741.

Comment by Vijay Prakash [03/Jun/21]

Command used to setup build environment for Fluoride BT stack at https://android.googlesource.com/platform/system/bt/+/master

```
$ mkdir -p ~/.bin
$ PATH="${HOME}/.bin:${PATH}"
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/.bin/repo
$ chmod a+rx ~/.bin/repo
```

```
sudo apt-get install git-core gnupg flex bison gperf build-essential   zip curl zlib1g-dev gcc-multilib g++-multilib   x11proto-core-dev libx11-dev lib32z-dev libncurses5   libgl1-mesa-dev libxml2-utils xsltproc unzip liblz4-tool libssl-dev
libc++-dev libevent-dev   flatbuffers-compiler libflatbuffers1   openssl libssl-dev
```

Comment by Vijay Prakash [07/Jun/21]

Tried building the Fluoride BT stack, but couldn't due to compilation errors. First, tried building the stack as a static library - it gave compilation error at the rust stage. Then tried building it as a shared library, libbluetooth.so, but got an error at the main stage.

Comment by Dan Regalado (Inactive) [08/Jun/21]

Vijay Prakash please paste here the full run to see if can spot something
cc:Ruian Duan

Comment by Ruian Duan [08/Jun/21]

Hi Vijay Prakash, I think the issue might be that the Fluoride BT won't build by itself. We need to build the whole AOSP. When doing CodeQL scan, what I did is (1) build the whole AOSP, (2) remove the output files related to BT, (3) rerun the build command with CodeQL, which will only build the missing output files, i.e. BT.

Let me also have a look and will update here.

Comment by Vijay Prakash [09/Jun/21]

Finally, I was able to build the Fluoride BT stack without AOSP after learning gn build system and fixing few errors in the source code, GN build files, and python build file. Changes I had to do are below found with `git diff` command:

```
diff --git a/build.py b/build.py
index 808c80891..784ea9ffc 100755
--- a/build.py
+++ b/build.py
@@ -34,6 +34,7 @@ import shutil
 import six
 import subprocess
 import sys
+import time

 # Use flags required by common-mk (find -type f | grep -nE 'use[.]' {})
 COMMON_MK_USES = [
@@ -139,7 +140,8 @@ class HostBuild():
         # Unless set, always build test code
         if not self.args.notest:
             target_use.append('test')
-
+        target_use.append('bt_dynlib')
+        #target_use.append('android')
         self.use = UseFlags(target_use)

         # Validate platform directory
```

Comment by Vijay Prakash [10/Jun/21]

I also found an android application that uses NDK and I was able to build and run it.

The next step is using the native application example, create a Bluetooth server that uses the Android BT shared library (libbluetooth.so), and try to trigger the bug. One possible issue that might occur is this libbluetooth.so was build for Linux and the Android application is going to run on Android, so it might need libbluetooth.so built with Android flag.

Comment by Vijay Prakash [10/Jun/21]

Another way to check the issue is by testing the issue against the Fluoride BT stack for Linux. I have libbluetooth.so for Linux, I need to compile the BT server and a client for Linux and see if we can get crash.

Comment by Vijay Prakash [10/Jun/21]

https://stackoverflow.com/questions/30813854/how-do-bluetooth-sdp-and-uuids-work-specifically-for-android

Comment by Vijay Prakash [14/Jun/21]

I was able to build Fluoride BT client and server for Linux. Build failure issue is resolved after upgrading libprotobuf-dev from 3.6 to 3.12; 3.6 is the highest version available on Ubuntu 20.04, I had to add Debian repo to my Ubuntu box.

Comment by Vijay Prakash [16/Jun/21]

Code flow of BT SDP in Android native application

https://android.googlesource.com/platform/packages/apps/Bluetooth/+/refs/heads/master/jni/com_android_bluetooth.h#129

```
const bt_interface_t* getBluetoothInterface();
```

https://android.googlesource.com/platform/packages/apps/Bluetooth/+/refs/heads/master/jni/com_android_bluetooth_btservice_AdapterService.cpp#94

```
const bt_interface_t* getBluetoothInterface() { return sBluetoothInterface; }
```

https://android.googlesource.com/platform/packages/apps/Bluetooth/+/refs/heads/master/jni/com_android_bluetooth_btservice_AdapterService.cpp#83

```
static const bt_interface_t* sBluetoothInterface = NULL;
```

https://android.googlesource.com/platform/packages/apps/Bluetooth/+/refs/heads/master/jni/com_android_bluetooth_btservice_AdapterService.cpp#825

```
  if (hal_util_load_bt_library((bt_interface_t const**)&sBluetoothInterface)) {
    ALOGE("No Bluetooth Library found");
  }
```

https://android.googlesource.com/platform/packages/apps/Bluetooth/+/refs/heads/master/jni/com_android_bluetooth_btservice_AdapterService.cpp#741

```
int hal_util_load_bt_library(const bt_interface_t** interface) {
  const char* sym = BLUETOOTH_INTERFACE_STRING;
  bt_interface_t* itf = nullptr;
  // The library name is not set by default, so the preset library name is used.
  void* handle = dlopen("libbluetooth.so", RTLD_NOW);
  if (!handle) {
    const char* err_str = dlerror();
    ALOGE("%s: failed to load Bluetooth library, error=%s", __func__,
          err_str ? err_str : "error unknown");
    goto error;
  }
  // Get the address of the bt_interface_t.
  itf = (bt_interface_t*)dlsym(handle, sym);
  if (!itf) {
    ALOGE("%s: failed to load symbol from Bluetooth library %s", __func__, sym);
    goto error;
  }
  // Success.
  ALOGI("%s: loaded Bluetooth library successfully", __func__);
  *interface = itf;
  return 0;
error:
```

```
vprakash@zmas:/home/rduan/android/source/system/bt$ grep -rn BLUETOOTH_INTERFACE_STRING . --exclude-dir=cpp-db --exclude-dir=bt-cpp-db
./btcore/src/hal_util.cc:36:  const char* sym = BLUETOOTH_INTERFACE_STRING;
./include/hardware/bluetooth.h:733:#define BLUETOOTH_INTERFACE_STRING "bluetoothInterface"
```

./bt/btif/src/bluetooth.cc:560

```
    create_bond_out_of_band,
    remove_bond,
    cancel_bond,
    get_connection_state,
    pin_reply,
    ssp_reply,
    get_profile_interface,
    dut_mode_configure,
    dut_mode_send,
    le_test_mode,
    set_os_callouts,
    read_energy_info,
    dump,
    dumpMetrics,
    config_clear,
    interop_database_clear,
    interop_database_add,
    get_avrcp_service,
    obfuscate_address,
    get_metric_id,
    set_dynamic_audio_buffer_size,
};
```

===

https://android.googlesource.com/platform/packages/apps/Bluetooth/+/refs/heads/master/jni/com_android_bluetooth_sdp.cpp#56

```
  }
  if (sBluetoothSdpInterface != NULL) {
    ALOGW("Cleaning up Bluetooth SDP Interface before initializing...");
    sBluetoothSdpInterface->deinit();
    sBluetoothSdpInterface = NULL;
  }
  sBluetoothSdpInterface = (btsdp_interface_t*)btInf->get_profile_interface(
      BT_PROFILE_SDP_CLIENT_ID);
  if (sBluetoothSdpInterface == NULL) {
    ALOGE("Error getting SDP client interface");
  } else {
    sBluetoothSdpInterface->init(&sBluetoothSdpCallbacks);
  }
  sCallbacksObj = env->NewGlobalRef(object);
}

bt/btif/src/bluetooth.cc

125 extern const btsdp_interface_t* btif_sdp_get_interface();

432   if (is_profile(profile_id, BT_PROFILE_SDP_CLIENT_ID))
433     return btif_sdp_get_interface();
```

bt/btif/src/btif_sdp.cc

```
144 static const btsdp_interface_t sdp_if = {
145     sizeof(btsdp_interface_t), init, deinit, search, create_sdp_record,
146     remove_sdp_record};
147
148 const btsdp_interface_t* btif_sdp_get_interface(void) {
149   BTIF_TRACE_DEBUG("%s", __func__);
150   return &sdp_if;
151 }
```

Comment by Vijay Prakash [17/Jun/21]

BT device seems to be virtualized in Android emulator. I am going to try to check the code against Android device first because on Linux VM I don't have actual BT device.

Comment by Vijay Prakash [17/Jun/21]

Also, from a conversation with the developer of Fluoride BT stack for Linux at Google - "The machine would need a Bluetooth device to run the build binary and library, at the moment it defaults to hci0. Documentation in details will come in next few months." It was my shortsightedness that I didn't consider that my VM will not have a Bluetooth device before heading on the Linux path.

Comment by Vijay Prakash [22/Jun/21]

I wrote a native android app using the native-activity samples available at Github. libbluetooth.so is not available to be loaded dynamically using `dlopen` in Android. My changes to native-activity app are below:

```
diff --git a/native-activity/app/build.gradle b/native-activity/app/build.gradle
index 8366a55..e6f3583 100644
--- a/native-activity/app/build.gradle
+++ b/native-activity/app/build.gradle
@@ -2,7 +2,7 @@ apply plugin: 'com.android.application'

 android {
     compileSdkVersion 29
-    ndkVersion '21.2.6472646'
+    ndkVersion '22.1.7171670'

     defaultConfig {
         applicationId = 'com.example.native_activity'
diff --git a/native-activity/app/src/main/cpp/CMakeLists.txt b/native-activity/app/src/main/cpp/CMakeLists.txt
index 2ecfd9e..cdaeeb9 100644
--- a/native-activity/app/src/main/cpp/CMakeLists.txt
+++ b/native-activity/app/src/main/cpp/CMakeLists.txt
@@ -16,13 +16,20 @@

 cmake_minimum_required(VERSION 3.4.1)

+#include Fluoride BT stack
+include_directories(/home/stoic/fluoride/bt/include)
```

Comment by Vijay Prakash [22/Jun/21]

I tried manually adding libbluetooth.so to the app and loading it with dlopen but couldn't succeed with changes below:

```
diff --git a/native-activity/app/build.gradle b/native-activity/app/build.gradle
index 8366a55..8aa7005 100644
--- a/native-activity/app/build.gradle
+++ b/native-activity/app/build.gradle
@@ -2,7 +2,7 @@ apply plugin: 'com.android.application'

 android {
     compileSdkVersion 29
-    ndkVersion '21.2.6472646'
+    ndkVersion '22.1.7171670'

     defaultConfig {
         applicationId = 'com.example.native_activity'
@@ -13,6 +13,9 @@ android {
                 arguments '-DANDROID_STL=c++_static'
             }
         }
+        ndk {
+            abiFilters 'x86_64'
+        }
     }
     buildTypes {
         release {
```

Seems like I will have to compile libbluetooth.so for Android because this post says binaries and shared libs compiled couldn't be used on Android easily.

Another thing I wanna try is to use the static library of the BT stack, which would require building my application's native library as a static library.

Comment by Vijay Prakash [24/Jun/21]

I tried building the app with static library but static library doesn't expose the APIs needed for SDP calls.

```
extern bt_interface_t bluetoothInterface;
static const bt_interface_t* sBluetoothInterface = NULL;
static const btsdp_interface_t* sBluetoothSdpInterface = NULL;


int hal_util_static_bt_library() {
    bt_interface_t *itf = &bluetoothInterface;
    sBluetoothSdpInterface = (btsdp_interface_t*)itf->get_profile_interface(
            BT_PROFILE_SDP_CLIENT_ID);
    LOGI("SDP intferace found");
}
```

Changes in the CMakeLists.txt file:

```
target_link_libraries(native-activity
    android
    native_app_glue
    EGL
    GLESv1_CM
    log
    /home/stoic/fluoride/bt/output_dir_rust/out/Default/libbluetooth.a)
```

Got linker error:

```
ld: error: undefined symbol: bluetoothInterface
```

Contents of libbluetooth.a:

```
~/fluoride/bt$ ar -t output_dir_rust/out/Default/libbluetooth.a
output_dir_rust/out/Default/obj/bt/main/libbluetooth.bte_conf.o
output_dir_rust/out/Default/obj/bt/main/libbluetooth.bte_init_cpp_logging.o
output_dir_rust/out/Default/obj/bt/main/libbluetooth.bte_logmsg.o
output_dir_rust/out/Default/obj/bt/main/libbluetooth.bte_main.o
output_dir_rust/out/Default/obj/bt/main/libbluetooth.stack_config.o
```

Comment by Vijay Prakash [02/Jul/21]

Different ways I tried so to use libbluetooth.so/libbluetooth.a (and other static libraries) build for linux intel x86-64 arch in Android app:

1. Use the libbluetooth.so shared library in Android device. It didn't work with the ARM devices because of the wrong architecture. Then I tried it with Intel device but didn't work because of runtime issue:

   ```
   native_activity-tUlGAy3maHm0sPNcATib4w==/lib/x86_64/libnative-activity.so": dlopen failed: library "libdl.so.2" not found: needed by /data/app/~~BxanCS2p8eC3zJw-etVc0w==/com.example.native_activity-
   tUlGAy3maHm0sPNcATib4w==/lib/x86_64/libbluetooth.so in namespace classloader-namespace
   ```

2. Use the libbluetooth.a and all the other static libraries. It didn't work because of a runtime error - librt.so couldn't be found.

I tired using libluetooth.so built for x86 ARM architecture but it didn't work on `arm64-v8a` and `armeabi-v7a` arm devices due to compatibility issues. And there are no 32-bit devices. Exact error I got for ARM devices is:

```
libbluetooth.so is incompatible with aarch64linux
```

I will try building libbluetooth from AOSP project for 64 bit ARM architecture and see if I can use it.

Comment by Vijay Prakash [02/Jul/21]

Different ABI that can be used:

```
arm64-v8a
armeabi
```

```
armeabi-v7a
mips
x86
x86_64
```

Comment by Vijay Prakash [14/Jul/21]

Tried building the app for x86_64 with libbluetooth.so build for x86_64.

During the launch of the multiple .so were needed. All of them were available in the AOSP build and including them in the app solved the problem except for the `android.hardware.bluetooth.a2dp@1.0.so`. Even after adding the this .so in the app it couldn't be located by the app.

```
ls lib/x86_64/
android.hardware.bluetooth.a2dp@1.0.so  libbluetooth.so  libchrome.so  libgrpc++.so  libgrpc_wrap.so  libnative-activity.so  libstatslog.so
```

Error at runtime:

```
/data/app/~~J6dVW4LGncsbvMxzwtgq7w==/com.example.native_activity-Je33LNoodZDV37TTvXSpEw==/lib/x86_64/libnative-activity.so": dlopen failed: library "android.hardware.bluetooth.a2dp@1.0.so" not found: needed by
/data/app/~~J6dVW4LGncsbvMxzwtgq7w==/com.example.native_activity-Je33LNoodZDV37TTvXSpEw==/lib/x86_64/libnative-activity.so in namespace classloader-namespace
```

Comment by Ruian Duan [14/Jul/21]

Hi Vijay Prakash, I searched the error message "namespace classloader-namespace" and found the following post. It seems relevant. According to the answers, it's an issue from Nougat onwards. So in addition to the two options listed by the answer, an alternative in our use case is to use versions lower than Nougat.

https://stackoverflow.com/questions/59608865/library-is-not-accessible-for-the-namespace-classloader-namespace

Comment by Vijay Prakash [15/Jul/21]

I was able to build the app with libbluetooth.so and it's dependent shared objects (so), and able to load libbluetooth.so with
{dlopen}

in the app. The issue in my previous comment is that android linker doesn't support versioned shared objects, see this issue https://stackoverflow.com/questions/11491065/linking-with-versioned-shared-library-in-android-ndk.

Apart from versioning, android also doesn't support `@` character in library names. Hack I did was to rename all the libraries to remove the versioning from the file and at all the places where this library was used in other libraries.

To solve the issue I used this script:

```
echo "" > change.log
for var in `ls -1`;do
        echo $var
        sed -i "s/@\([0-9]\).\([0-9]\).so/_\1_\2.so/g" $var
        new_file=`echo $var|sed "s/@\([0-9]\).\([0-9]\).so$/_\1_\2.so/g"`
        mv $var $new_file
        objdump -p $new_file | grep so >> change.log
done
```

Copy all the libraries in a directory and run this script. What it does is replace the version like `mylibrarry@1.3.so` with `mylibrary_1_3.so` where it's used as a dependency in a library and rename the library file itself.

List of shared objects (so/libraries) I had to add in the CMakeLists.txt is below:

```
diff --git a/native-activity/app/src/main/cpp/CMakeLists.txt b/native-activity/app/src/main/cpp/CMakeLists.txt
index 2ecfd9e..2520c9e 100644
--- a/native-activity/app/src/main/cpp/CMakeLists.txt
+++ b/native-activity/app/src/main/cpp/CMakeLists.txt
@@ -16,13 +16,18 @@

 cmake_minimum_required(VERSION 3.4.1)

+#include Fluoride BT stack
+include_directories(/home/stoic/fluoride/bt/include)
+include_directories(/usr/include/libchrome/)
+include_directories(/home/stoic/fluoride/bt/types)
+
 # build native_app_glue as a static lib
 set(${CMAKE_C_FLAGS}, "${CMAKE_C_FLAGS}")
 add_library(native_app_glue STATIC
     ${ANDROID_NDK}/sources/android/native_app_glue/android_native_app_glue.c)

 # now build app's shared lib
-set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=gnu++11 -Wall -Werror")
+set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -stdlib=libc++ -Wall")
```

In the build.gradle I had to specify that I want to build the app for x86_64 ABI:

```
diff --git a/native-activity/app/build.gradle b/native-activity/app/build.gradle
index 8366a55..c035fe1 100644
--- a/native-activity/app/build.gradle
+++ b/native-activity/app/build.gradle
@@ -2,17 +2,21 @@ apply plugin: 'com.android.application'

 android {
     compileSdkVersion 29
-    ndkVersion '21.2.6472646'
+    ndkVersion '22.1.7171670'

     defaultConfig {
         applicationId = 'com.example.native_activity'
         minSdkVersion 14
-        targetSdkVersion 28
+        targetSdkVersion 29
         externalNativeBuild {
             cmake {
                 arguments '-DANDROID_STL=c++_static'
             }
         }
+
```

To load the source code I added this function in native-activity app from Google available on Github:

```
diff --git a/native-activity/app/src/main/cpp/main.cpp b/native-activity/app/src/main/cpp/main.cpp
index fe34068..7b2e903 100644
--- a/native-activity/app/src/main/cpp/main.cpp
+++ b/native-activity/app/src/main/cpp/main.cpp
@@ -31,9 +31,18 @@
 #include <android/log.h>
 #include <android_native_app_glue.h>

+// Bluetooth related header files
+#include <hardware/bluetooth.h>
+#include <hardware/bt_sdp.h>
+#include <dlfcn.h>
+
 #define LOGI(...) ((void)__android_log_print(ANDROID_LOG_INFO, "native-activity", __VA_ARGS__))
 #define LOGW(...) ((void)__android_log_print(ANDROID_LOG_WARN, "native-activity", __VA_ARGS__))

+extern bt_interface_t bluetoothInterface;
+static const bt_interface_t* sBluetoothInterface = NULL;
+static const btsdp_interface_t* sBluetoothSdpInterface = NULL;
+
 /**
  * Our saved state data.
```

Comment by Vijay Prakash [19/Jul/21]

Couldn't initialize the BT stack. Got this error:

```
I/native-activity: =============loaded
    hal_util_load_bt_library: loaded Bluetooth library successfully
I/bt_btif: system/bt/btif/src/bluetooth.cc:156 init: init: start restricted = 1 ; common criteria mode = 0, config compare result = 0
I/bt: bt_common::init_flags: Flags loaded: gd_core=false gd_advertising=false gd_scanning=false gd_security=false gd_acl=false gd_l2cap=false gd_hci=false gd_controller=false gatt_robust_caching=false btaa_hci=false gd_rust=false
gd_link_policy=false
I/bt_osi_allocation_tracker: system/bt/osi/src/allocation_tracker.cc:59 allocation_tracker_init: canary initialized
I/native_activit: [0716/161204.285187:INFO:message_loop_thread.cc(224)] Run: message loop starting for thread bt_stack_manager_thread
I/bt_stack_manager: system/bt/btif/src/stack_manager.cc:200 event_init_stack: event_init_stack is initializing the stack
E/libc: Access denied finding property "persist.bluetooth.factoryreset"
E/native_activit: [0716/161204.359089:ERROR:config.cc(104)] config_new: unable to open file '/data/misc/bluedroid/bt_config.conf': Permission denied
W/bt_btif_config: system/bt/btif/src/btif_config.cc:272 init: init unable to load config file: /data/misc/bluedroid/bt_config.conf; using backup.
E/native_activit: [0716/161204.360478:ERROR:config.cc(104)] config_new: unable to open file '/data/misc/bluedroid/bt_config.bak': Permission denied
W/bt_btif_config: system/bt/btif/src/btif_config.cc:281 init: init unable to load backup; attempting to transcode legacy file.
E/bt_btif_config_transcode: system/bt/btif/src/btif_config_transcode.cc:33 btif_config_transcode: btif_config_transcode unable to load XML file '/data/misc/bluedroid/bt_config.xml': 3
E/bt_btif_config: system/bt/btif/src/btif_config.cc:288 init: init unable to transcode legacy file; creating empty config.
W/native_activit: [0716/161204.361959:WARNING:btif_config.cc(166)] read_or_set_metrics_salt: Failed to read metrics salt from config
I/native_activit: [0716/161204.362120:INFO:btif_config.cc(177)] read_or_set_metrics_salt: Metrics salt is not invalid, creating new one
W/libc: pthread_create sched_setscheduler(1, {1}) call failed: Operation not permitted
E/bt_osi_alarm: system/bt/osi/src/alarm.cc:683 timer_create_internal: timer_create_internal unable to create timer with clock 7: Operation not permitted
A/native_activit: [0716/161204.373283:FATAL:alarm.cc(169)] Check failed: false.
    #00 0x00007836fb7f53d9 /data/app/~~Gke2_GXLDZI1kACsh2Yydg==/com.example.native_activity-FghcKa8ZBNkLkfkM-de9mA==/lib/x86_64/libchrome.so+0x00000000000c83d9
    #01 0x00007836fa91f09b /data/app/~~Gke2_GXLDZI1kACsh2Yydg==/com.example.native_activity-FghcKa8ZBNkLkfkM-de9mA==/lib/x86_64/libbluetooth.so+0x000000000042209b
```

I will try to fix this issue this week by trying to run the app with system privileges or running it android device where my will have access to the `/data/misc/bluedroid/bt_config.conf` and have permission to all the operation that gave error.

Comment by Vijay Prakash [06/Aug/21]

I tried to run the application as a system app but still couldn't fix the above issue.

I tried to run adb as root and change permission of /data/ directory in the emulator device. Reference for adb related commands can be found at https://developer.android.com/studio/command-line/adb.

Things I tried:

```
To run adb as root: Android/Sdk/platform-tools/adb root
To list the emulators: adb devices
To start a shell in the emulator device: Android/Sdk/platform-tools/adb shell #if there is only one emulator adb connects to that one automatically, else it needs to be told to which one to connect
```

/data directory is where system apps reside and it's not accessible without root permission, that's why we need to run adb as root. I changed the permission with read, write, and execute for everyone with the command :

```
generic_x86_64_arm64:/ # chmod -R 0777 data

generic_x86_64_arm64:/ # ls -lad /data
drwxrwxrwx 47 system system 4096 2021-07-14 11:18 /data
```

/data/app is where the system application resides.

I can see my applications in /data/app

```
ls -lah /data/app/~~Oxvve48otTGa2W1j6x6_4w\=\=/com.example.native_activity-Xb7e_8krfhwwgtEsB58h4Q\=\=/
total 4.5M
drwxrwxr-x 3 system system 4.0K 2021-08-02 22:36 .
drwxrwxr-x 3 system system 4.0K 2021-08-02 22:36 ..
-rw-r--r-- 1 system system 9.0M 2021-08-02 22:36 base.apk
drwxr-xr-x 3 system system 4.0K 2021-08-02 22:36 lib
```

Things I wanna try - to launch an app as root from inside the emulator shell and locate where libluetooth.so reside for Bluetooth of the device.

Comment by Ruian Duan [06/Aug/21]

Hi Vijay Prakash, per our discussion, just adding more contexts on my laptop setup here.

The **emulator** binary and the images maintained by **Android Studio** are shown as below. The commands to start emulator can be found in:

https://developer.android.com/studio/run/emulator-commandline
https://source.android.com/setup/create/avd

```
[rduan@macos ~/Library/Android/sdk]$ pwd
/Users/rduan/Library/Android/sdk
[rduan@macos ~/Library/Android/sdk]$ find . -name emulator
./tools/emulator
./system-images/android-30/google_apis/x86/data/misc/emulator
./emulator
./emulator/emulator
[rduan@macos ~/Library/Android/sdk]$ find . -name "*.img"
./system-images/android-30/google_apis/x86/encryptionkey.img
./system-images/android-30/google_apis/x86/ramdisk.img
./system-images/android-30/google_apis/x86/system.img
./system-images/android-30/google_apis/x86/vendor.img
./system-images/android-30/google_apis/x86/userdata.img
```

Comment by Vijay Prakash [09/Aug/21]

Last week we ran into a dead-end trying to initialize the BT stack manually by loading the libbluetooth.so in our application. As mentioned in the previous comment, the new approach would be to update the Bluetooth JNI in AOSP to include a change that could trigger the issue and build a new emulator and then try to trigger with an application. This way we don't have to go through the process of initializing the BT stack.

Comment by Vijay Prakash [20/Aug/21]

```
~/Library/Android/sdk/emulator/emulator -list-avds

~/Library/Android/sdk/emulator/emulator @Pixel_5_API_30 -system /Users/vprakash/Work/research/custom-android-images/system-v1.img -no-boot-anim -no-window


https://source.android.com/setup/build/building

https://source.android.com/setup/build/gsi#building-gsis

https://source.android.com/setup/build/gsi#flashing-gsis ( shows how to flash GSI, use system.img and vmmeta.img)

https://source.android.com/setup/build/running#unlocking-recent-devices

https://source.android.com/devices/bootloader/locking_unlocking


https://source.android.com/setup/create/avd

sudo vi ./source/packages/apps/Bluetooth/jni/com_android_bluetooth_sdp.cpp
source//system/bt//osi/src/allocation_tracker.cc
jni/com_android_bluetooth_sdp.cpp
source/system/bt/btif/src/btif_sdp_server.cc
source/system/bt/osi/src/allocator.cc
```

==
crash log:

```
08-18 19:28:16.625  5463  5463 D ObexServerSockets2: startAccept()
08-18 19:28:16.625  5463  5463 D BluetoothSdpJni: =====sdpCreateOppOpsRecordNative
08-18 19:28:16.626  5463  5463 I BluetoothSdpJni: =====bluetooth_sdp_record size: 96 or 0x00000000000060
08-18 19:28:16.626  5463  5463 I BluetoothSdpJni: =====name_str: OBEX Object Push
08-18 19:28:16.626  5463  5463 I BluetoothSdpJni: =====SDP Create record with original service name: OBEX Object Push
08-18 19:28:16.626  5463  5463 I BluetoothSdpJni: =====record.ops.hdr.service_name_length: 4294967185 or 0xffffff91
08-18 19:28:16.626  5463  5463 I BluetoothSdpJni: =====record.hdr.service_name_length: 4294967185 or 0xffffff91
08-18 19:28:16.626  5463  5463 I BluetoothSdpJni: =====Goint to call create_sdp_record
08-18 19:28:16.626  5463  5463 I bt_btif_sdp_server: system/bt/btif/src/btif_sdp_server.cc:126 get_sdp_records_size: size_of records_size: 4
08-18 19:28:16.626  5463  5463 I bt_btif_sdp_server: system/bt/btif/src/btif_sdp_server.cc:135 get_sdp_records_size: record_size: -14, 0x00000000fffffff2
08-18 19:28:16.626  5463  5463 I bt_btif_sdp_server: system/bt/btif/src/btif_sdp_server.cc:138 get_sdp_records_size: record_size: -14, 0x00000000fffffff2
08-18 19:28:16.626  5463  5463 I bt_btif_sdp_server: system/bt/btif/src/btif_sdp_server.cc:199 alloc_sdp_slot: =====calculated size of in_record: 0x00000000fffffff2
08-18 19:28:16.626  5463  5463 I bt_os_allocator: system/bt/osi/src/allocator.cc:64 osi_malloc: =====requested allocation size: 18446744073709551602 or 0xfffffffffffffff2
08-18 19:28:16.626  5463  5463 I bt_os_allocator: system/bt/osi/src/allocator.cc:67 osi_malloc: =====real size allocated after allocation tracker resize for canary: 2 or 0x00000000000002
08-18 19:28:16.626  5463  5463 I bt_os_allocator: system/bt/osi/src/allocator.cc:70 osi_malloc: =====going to notify alloc tracker
08-18 19:28:16.626  5463  5463 I bt_btif_sdp_server: system/bt/btif/src/btif_sdp_server.cc:201 alloc_sdp_slot: =====going to copy input record from 0x7fff16a37110 to 0x705df44758b8
08-18 19:28:16.626  5463  5463 I bt_btif_sdp_server: system/bt/btif/src/btif_sdp_server.cc:202 alloc_sdp_slot: =====at least required size of in_record: 0x0000000000000060
08-18 19:28:16.626  5463  5463 F libc    : Fatal signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0x705e044e7000 in tid 5463 (droid.bluetooth), pid 5463 (droid.bluetooth)
08-18 19:28:16.631  5463  5525 D ObexServerSockets2: Accepting socket connection...
08-18 19:28:16.632  5463  5524 D ObexServerSockets2: Accepting socket connection...
08-18 19:28:16.651  5528  5528 I crash_dump64: obtaining output fd from tombstoned, type: kDebuggerdTombstoneProto
08-18 19:28:16.652   220   220 I tombstoned: received crash request for pid 5463
08-18 19:28:16.652  5528  5528 I crash_dump64: performing dump of process 5463 (target tid = 5463)
```

Generated at Fri Aug 20 18:18:44 UTC 2021 by Vijay Prakash using Jira 8.13.7#813007-sha1:3e6833b6aa4b98966fa10c8ab5e2a573d1da0bde.