天猫精灵 **x1 + ESP8266** 模拟 **Broadlink SP Mini** 模拟尝试：**BroadLink** 协议篇 | 天猫精灵开发

Phodal Huang

September 8, 2017

目录

原文链接:https://www.wandianshenme.com/play/aligenie-tmall-x1-broadlink-spmini-emulator-ana

因为天猫精灵 x1 目前只能支持 Broadlink 设备，便想着是不是能自己做一个设备来模拟。这样一来，我们就可以绕过服务端，从而便可以直接通过天猫精灵来控制本地的设备。

为此便开始了一系列的计划，即两步曲:

- 分析 broadlink 的通信协议
- 模拟 broadlink 设备

本文则是 python-broadlink 项目中，对于协议解析的翻译，原文地址是：Broadlink RM2 network protocol

加密

Broaalink 数据包中，包括了基于 CBC 模式的 AES 加密。初始的键（key）是:

```
0x09, 0x76, 0x28, 0x34, 0x3f, 0xe9, 0x9e, 0x23, 0x76, 0x5c, 0x15, 0x13,
    0xac, 0xcf, 0x8b, 0x02
```

其中的 IV 值是:

```
0x56, 0x2e, 0x17, 0x99, 0x6d, 0x09, 0x3d, 0x28, 0xdd, 0xb3, 0xba, 0x69,
    0x5a, 0x2e, 0x6f, 0x58
```

其 Python 代码如下所示:

```
self.key = bytearray([0x09, 0x76, 0x28, 0x34, 0x3f, 0xe9, 0x9e, 0x23, 0x76,
    0x5c, 0x15, 0x13, 0xac, 0xcf, 0x8b, 0x02])
self.iv = bytearray([0x56, 0x2e, 0x17, 0x99, 0x6d, 0x09, 0x3d, 0x28, 0xdd,
    0xb3, 0xba, 0x69, 0x5a, 0x2e, 0x6f, 0x58])
```

校验（**Checksum**）

构造数据包，并将校验字节设置为零。并将每个字节添起始值 oxbeaf，同时再用 oxffff 包装。

示例代码，如下所示:

```
checksum = 0xbeaf
checksum += payload
```

```
3 checksum = checksum & 0xffff
```

新设备设置

要在 AP 模式下，设置新的 Broadlink 设备。则需要将 136 字节的数据包发送到设备，格式如下所示：

Offset	内容
0x00-0x19	00
0x20-0x21	校验作为一个小端 16 位整数（little-endian 16 bit integer）
0x26	14 (总是 14)
0x44-0x63	SSID 名字 (剩余的部分使用 0 填充)
0x64-0x83	密码 (剩余的部分使用 0 填充)
0x84	SSID 的字符长度
0x85	密码的字符长度
0x86	无线网络的安全模式 (00 - none, 01 = WEP, 02 = WPA1, 03 = WPA2, 04 = WPA1/2)
0x87-88	00

将此数据包作为 UDP 广播，发送到 255.255.255.255 上的 80 端口。

网络发现

要发现本地网络上的 Broadlink 设备，请发送一个包含以下内容的 48 字节数据包：

Offset	内容
0x00-0x07	00
0x08-0x0b	以当前 GMT 的偏移值作为一个小端（little-endian）32 位整数
0x0c-0x0d	以当前的年一个小端 16 位整数
0x0e	当前的小时数分钟

Offset	内容
0x0f	当前午夜的小时数
0x10	本世纪现在的几年
0x11	今天在本周是第几天 (Monday = 0, Tuesday = 1, etc)
0x12	今天在本月是第几天
0x13	本月是第几月
0x19-0x1b	本地 IP 地址
0x1c-0x1d	源端口作为一个小端 16 位整数
0x1e-0x1f	00
0x20-0x21	校验作为一个小端 16 位整数
0x22-0x25	00
0x26	06
0x27-0x2f	00

将此数据包作为 UDP 广播，发送到 255.255.255.255 上的 80 端口。

对应广播代码如下所示：

```
cs = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
cs.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
cs.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
cs.bind((local_ip_address,0))
```

```
5 cs.sendto(packet, ('255.255.255.255', 80))
```

响应（任何单播响应）：

Offset	内容
0x34-0x35	设备类型为小端 16 位整数（请参见设备类型映射）
0x3a-0x40	目标设备的 MAC 地址

以下是一个处理响应的示例代码：

```
1 response = cs.recvfrom(1024)
2 responsepacket = bytearray(response[0])
3 host = response[1]
4 mac = responsepacket[0x3a:0x40]
5 devtype = responsepacket[0x34] | responsepacket[0x35] << 8
6 return gendevice(devtype, host, mac)
```

设备类型映射：

在响应包中的设备类型	设备精英	视为
0	SP1	SP1
0x2711	SP2	SP2
0x2719 or 0x7919 or 0x271a or 0x791a	Honeywell SP2	SP2
0x2720	SPMini	SP2
0x753e	SP3	SP2
0x2728	SPMini2	SP2
0x2733 or 0x273e	OEM branded SPMini	SP2
>= 0x7530 and <= 0x7918	OEM branded SPMini2	SP2
0x2736	SPMiniPlus	SP2
0x2712	RM2	RM
0x2737	RM Mini / RM3 Mini Blackbean	RM
0x273d	RM Pro Phicomm	RM
0x2783	RM2 Home Plus	RM
0x277c	RM2 Home Plus GDT	RM
0x272a	RM2 Pro Plus	RM
0x2787	RM2 Pro Plus2	RM
0x278b	RM2 Pro Plus BL	RM

在响应包中的设备类型	设备精英	视为
0x278f	RM Mini Shate	RM
0x2714	A1	A1
0x4EB5	MP1	MP1

命令包格式

命令包 header 长为 56 字节，格式如下：

Offset	Contents
0x00	0x5a
0x01	0xa5
0x02	0xaa
0x03	0x55
0x04	0x5a
0x05	0xa5
0x06	0xaa
0x07	0x55

Offset	Contents
0x08-0x1f	00
0x20-0x21	全分组校验作为一个小端 16 位整数
0x22-0x23	00
0x24	0x2a
0x25	0x27
0x26-0x27	命令码作为一个小端 16 位整数
0x28-0x29	数据包计数为一个小端 16 位整数
0x2a-0x2f	本地 MAC 地址
0x30-0x33	本地设备 ID（认证期间获取，认证前为 00）
0x34-0x35	数据包头的校验和，作为一个小端 16 位整数
0x36-0x37	00

负载（payload）是在这之后立即添加进行的。在附加有效载荷之前，校验和的计算在 0x34；并且只覆盖 header。在负载（payload）添加之后，在 0x20 的校验将会计算。并覆盖整个数据包（包括 0x34 处的校验）。因此：

1. 生成校验和值设置为 0 的数据包头

2. 将校验的初始化值设置为 0xbeaf，并计算数据包头的校验和。将 0x34-0x35 设置为此值。
3. 附加有效载荷
4. 设置校验

授权

您必须从设备获取授权密钥才能进行通信。为此，生成一个包含以下内容的 80 字节数据包：

Offset	内容
0x00-0x03	00
0x04-0x12	表示此设备的 15 位数值。Broadlink 在实现中使用的是 IMEI
0x13	01
0x14-0x2c	00
0x2d	0x01
0x30-0x7f	包含设备名称的 NULL 终止 ASCII 字符串

使用带有命令值 0x0065 的包发送此有效载荷。响应数据包将包含从 0x38 开始的加密过的有效载荷。使用默认密钥和 IV 解密。解密的有效载荷的格式是：

Offset	内容
0x00-0x03	Device ID
0x04-0x13	Device 密钥

所有其他命令数据包，必须使用此加密密钥和设备 ID。

```
1 response = self.send_packet(0x65, payload)
2 payload = self.decrypt(response[0x38:])
3
4 self.id = payload[0x00:0x04]
5 self.key = payload[0x04:0x14]
```

进入学习模式

发送以下 16 字节有效负载，包含命令值为 0x006a：

Offset	内容
0x00	0x03
0x01-0x0f	0x00

代码表示如下：

```
def enter_learning(self):
  packet = bytearray(16)
  packet[0] = 3
  self.send_packet(0x6a, packet)
```

从学习模式读回数据

发送以下 16 字节有效负载，包含命令值为 0x006a：

Offset	内容
0x00	0x04
0x01-0x0f	0x00

响应的字节 0x22 包含一个小端 16 位错误代码。如果它是 0，则获得代码。字节 0x38 和之前的响应是被加密的。需要解密他们。字节 0x04 和解密的有效载荷之后，则是所捕获的有效数据。

```
def check_data(self):
  packet = bytearray(16)
  packet[0] = 4
  response = self.send_packet(0x6a, packet)
  err = response[0x22] | (response[0x23] << 8)
  if err == 0:
    payload = self.decrypt(bytes(response[0x38:]))
    return payload[0x04:]
```

发送数据

发送以下 16 字节有效负载，包含命令值为 0x006a：

Offset	内容
0x00	0x02
0x01-0x03	0x00
0x04	0x26 = IR, 0xb2 用于 RF 433Mhz, 0xd7 用于 RF 315Mhz
0x05	重复次数, (0 = 不重复, 1 重复两次, ...)
0x06-0x07	以下数据的长度为小端格式
0x08	脉冲长度为 32,84 ms（ms * 269/8192）
....	0x0d 0x05 结尾则表示 IR

每个值由一个字节表示。如果长度超过一个字节，那么它将以 0 开始来存储大端。

以下是 SP2 设置电源状态的代码：

```
def set_power(self, state):
  """Sets the power state of the smart plug."""
  packet = bytearray(16)
  packet[0] = 2
  packet[4] = 1 if state else 0
  self.send_packet(0x6a, packet)
```

示例：我的 Optoma 投影机的头是 8920 4450

8920 * 269 / 8192 = 0x124 4450 * 269 / 8192 = 0x92

所以数据以 0x00 0x1 0x24 0x92 .... 开始

原文链接:https://www.wandianshenme.com/play/aligenie-tmall-x1-broadlink-spmini-emulator-ana