# iOS设备WiFi芯片受高危漏洞威胁 逾9成iOS用户受影响！

原创 2017-09-30 点击关注→ 百度安全实验室

## 1. 摘要

随着iOS 11的发布，多个BroadCom WiFi芯片的高危漏洞被公开[1]。这些漏洞对上亿台未来得及更新的iOS设备造成了严重的安全威胁。黑客可对同一WiFi网络下的设备发起攻击，远程控制受害设备的WiFi芯片，甚至进一步攻破iOS内核。本文对iOS设备WiFI芯片相关漏洞进行简要技术分析，然后根据iOS设备的系统版本统计出受影响的规模。截至9月27日，国内92.3%的iOS用户都受到相关高危漏洞的威胁。我们呼吁用户尽快升级iOS系统到最新版本，并号召手机厂商采用更有效的防护技术，避免用户受到已知高危漏洞的威胁。

## 2. BroadCom WiFi芯片漏洞技术分析

本文着重分析两个BroadCom WiFi芯片漏洞：CVE-2017-11120和CVE-2017-11121。这两个漏洞都是WiFi芯片固件代码在处理数据帧时缺乏对特定字段的严格校验。攻击者可以利用它们制造内存破坏，实现任意代码执行，获得对设备的远程控制。

### 2.1 漏洞CVE-2017-11120

iOS设备搭载的BroadCom WiFi芯片采用了快速基本服务设置转换（Fast BSS Transition）和无线资源管理（Radio Resource Management）标准。在接入无线接入点（Access Point，简称AP）后，iOS设备会发送相邻接入点请求（Neighbor Report Request），AP则返回相邻接入点应答（Neighbor Report Response），包含当前无线局域网内的相邻AP以及各自的BSSID，Operating Class和Channel Number等信息。在处理Neighbor Report Response数据帧时，BroadCom WiFi芯片将每一种Operating Class和Neighbor Report信息保存在一个456字节的内存块中（如图1所示），并且将这些块通过指针串接起来。其中，Neighbor Count Array记录了各个Channel Number的Neighbor数量。Array长度为450字节，每2个字节记录一个Channel Number，所以最大可记录的Channel Number为224（0xE0）。

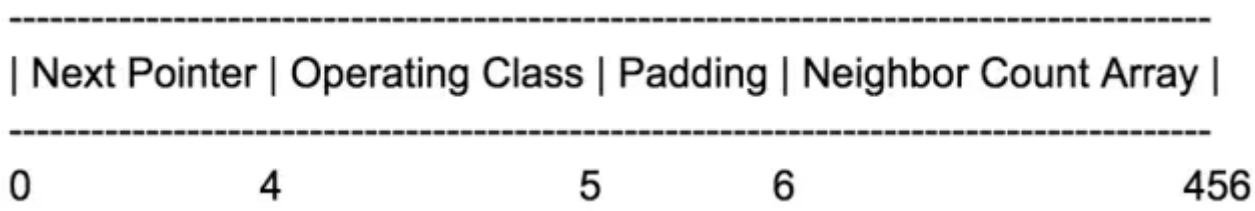


图1：BroadCom WiFi芯片记录Neighbor Report Response信息的内存块结构 [2]

```
int function_AC0A8(..., uint8_t* nrrep_buffer, ...) {
  ...
  //Find and increment neighbor in given channel for given OP-Class
  int res = function_AC07C(..., nrrep_buffer, ...);

  //If there's no entry for the given OP-Class, create and populate it
  if (!res) {
    uint8_t* buffer = malloc(456);
    if ( !buffer ) {
      ...
    }
    else {
      buffer[4] = nrrep_buffer[16];              //Operating Class
      uint8_t channel_number = nrrep_buffer[17]; //Channel Number
      uint16_t* chan_neighbor_count_arr = (uint16_t*)(buffer + 6);
      chan_neighbor_count_arr[channel_number]++;
      ...
    }
  }
  ...
}
```

图2：BroadCom WiFi芯片处理Neighbor Report Response信息的函数 [2]

如图2所示，WiFi芯片固件里位于地址0xAC0A8的函数（简称function_AC0A8，下同）首先在串联的内存块中查找Operating Class对应的条目，如果没有找到，则会动态创建一个条目 ，然后从数据帧中读出Channel Number作为数组索引，定位到Neighbor Count Array中元素，将其值加一。此过程并没有对Channel Number做出校验，当伪造的数据帧中Channel Number大于0xE0（如0xFF）时，上述过程将会造成内存越界写。攻击者可以通过内存越界写改变关键指针数据或者控制流元素，一步步接管代码执行。值得一提的是，BroadCom WiFi芯片里没有ASLR、DEP等防护，攻击者可以很容易做代码改写，注入任意代码执行。

此漏洞影响iOS 11.0以前的设备。目前此漏洞的利用代码已公开[2]，攻击者能直接复用这一利用代码攻击漏洞设备，在WiFi芯片中插入后门，并在用户无感知的情况下实现对设备的远程控制。

**2.2 漏洞CVE-2017-11121**

根据Fast BSS Transition标准，当设备在无线网络环境下进行快速漫游（fast roaming）时，会触发校验和重关联（reassociation）操作。Reassociation操作会对组临时秘钥（Group Temporal Key，简称GTK）进行解密和安装。这两个过程中存在多处memcpy调用，调用前都缺少对copy长度的校验，可能导致内存破坏。

如图3所示，reassociation由function_8462C函数负责，它调用function_6D8对GTK解密，然后会继续调用function_C9C14对GTK进行安装。相关代码片段如下：

```
int function_8462C(...) {
    ...

    //Getting the FT-IE
    uint8_t* ft_ie = bcm_parse_tlvs(..., ..., 55);
    if (!ft_ie)
        return 0;

    //Getting the GTK Sub-Element
    uint8_t* gtk_subelem = bcm_parse_tlvs(ft_ie + 84, ft_ie[1] - 82, 2);
    if (!gtk_subelem)
        return 0;
    ...

    //Decrypting the GTK
    context_struct->gtk_length = gtk_subelem[4];
    if (!function_6D8(16, ..., gtk_subelem[1] - 11, gtk_subelem + 13, &context_struct->decrypted_gt

        //Installing the GTK
        function_C9C14(..., context_struct->decrypted_gtk, context_struct->gtk_length, ...);
        ...
    }
    ...
}

function_6D8(unsigned key_length, char* key, unsigned input_length, char* input, char* output) {
    ...
    char buf[0x200];
    char buf2[0x8];

    //Validating the lengths
    if ( (key_length - 16) > 16 ||
        ((0x80808000 << (key_length - 16)) & 0x80000000) == 0 ||
        input_length > 0x188 ||
        input_length << 29 )
        return 1;

    //Copying the input into a local stack buffer
    memcpy(buf2, input, 8);
    memcpy(buf + 8, input + 8, input_length - 8);

    //Do AES decryption
    ...
}

int function_C9C14(..., char* gtk, int gtk_len, ...) {
    ...
    char* key_buffer = malloc(164);
    ...
    memcpy(key_buffer + 8, gtk, gtk_len);
    ...
}
```

图3：BroadCom WiFi芯片重关联操作时GTK解密和安装的相关函数 [3]

上述处理过程中，有两处memcpy调用存在问题：

1）GTK解密函数function_6D8中，当构造的畸形数据帧中gtk_subelem[1]为11时，函数参数input_length为0。在第二处调用memcpy时，input_length-8为0xfffffff8，这将导致大量数据被copy，破坏stack上的数据；

2）GTK安装函数function_C9C14，参数gtk_len取值为gtk_subelem[4]。攻击者可以构造畸形数据帧，使gtk_subelem[4]大于164，函数中memcpy调用前没有检查gtk_len取值，可能导致堆溢出。

攻击者同样可以攻击此漏洞造成内存破坏，实现远程任意代码执行。此漏洞影响系统版本在11.0以前的iOS设备。

## 3. 通过WiFi芯片漏洞可进一步攻击iOS内核

攻击者可将WiFi芯片漏洞作为跳板，进一步攻击iOS内核。iOS设备进行WiFi通信时，内核的WiFi驱动会向WiFi芯片发送ioctl请求。如果WiFi芯片被攻击者控制，攻击者能够篡改ioctl返回的结果数据，触发内核WiFi驱动中结果处理函数的漏洞，从而实现对iOS内核的攻击。

表1: 当攻击者控制WiFi芯片后，可用于攻击iOS内核驱动的漏洞

| 漏洞 | 影响iOS版本 | 漏洞危害 |
| --- | --- | --- |
| CVE-2017-7103[4] | <11.0 | 堆溢出，可能导致代码执行 |
| CVE-2017-7105[5] | <11.0 | 堆溢出，可能导致代码执行 |
| CVE-2017-7108[6] | <11.0 | 栈数据信息泄露 |
| CVE-2017-7110[7] | <11.0 | 堆溢出，可能导致代码执行 |
| CVE-2017-7112[8] | < 11.0 | 可控位置写入NULL byte |

表1中列举了可由WiFi芯片作为跳板攻击内核的漏洞。漏洞原理简要说明如下：

● CVE-2017-7103：驱动AppleBCMWLANBusInterfacePCIe中的函数completeFirmwareTimestampMsg，在Firmware Timestamp消息完成后会被回调。函数内部将Timestamp消息封装为mbuf，交由processFirmwareTimeSyncMessage处理，mbuf pkthdr_len设置为消息中timestamp_length字段的值。processFirmwareTimeSyncMessage函数内部存在一处memmove调用，长度参数为pkthdr_len。程序没有对pkthdr_len进行检查，构造过大的pkthdr_len会使memmove调用产生内存溢出。

● CVE-2017-7105：驱动AppleBCMWLANCore中的函数assembleBGScanResults，在处理WLC_GET_VAR ioctl返回的结果时，会调用IOMalloc分配一块堆内存，内存分配长度根据返回结果中的字段计算得出。代码中缺少对分配长度的溢出校验，在WiFi芯片被控制情况下，攻击者可通过篡改ioctl返回数据，使IOMalloc分配长度在计算时产生整型溢出，进而导致过小的内存分配，后续对分配内存的copy操作可能引起堆溢出。

● CVE-2017-7108：驱动AppleBCMWLANCore中的updateRateSetAsyncCallback函数，在处理WLC_GET_CURR_RATESET ioctl请求结果时，首先将0x14字节的结果数据中读到栈中buffer。rate数目由buffer中前4字节获得，接着函数从buffer+4处循环读出rate数据。由于在循环操作前缺少对rate数目的校验，攻击者可以篡改ioctl返回的rate结果，将rate数目字段改为过大的值，实现对buffer数据的越界读，造成栈上数据信息泄露。

● CVE-2017-7110：设备向WiFi芯片发送获得所有Vendor IE列表信息的ioctl请求，返回结果交由驱动AppleBCMWLANCore中setVendorIE函数处理。setVendorIE内部调用obvcopy时，length参数根据ioctl返

回结果中字段的计算得出。然而程序中缺乏对length值的校验，在WiFi芯片受控下，攻击者可以构造畸形请求结果，使得obvcopy的length过大，导致堆溢出。

● CVE-2017-7112：驱动AppleBCMWLANCore中handleTraceEvent函数，在处理WLC_E_TRACE消息时，缺少对消息头中len字段的检查。而后根据len计算得到的数组索引，进而改写数组内容时，可能对数组以外的内存写入NULL byte。

## 4. 绝大多数iOS用户受到WiFi高危漏洞威胁

截至2017年9月27日，百度安全实验室对国内上亿台iOS设备的系统版本进行了统计，排除虚假设备干扰结果后，详细系统版本比例分布如图4所示。从左半部分开始，逆时针方向按新旧版本次序依次为最新的iOS 11.x到iOS 7.x。其中，最新的iOS 11.x设备占比为7.7%，iOS 10.3.3系统占比50.8%，此版本之外的iOS 10.x版本（10.0.0至10.3.2）占比23.8%。早期的iOS 9.x设备占比11.2%，iOS 8.x 占比 5.8%，更早期的iOS 7占比0.7%。

从图中可以看到，目前国内升级到最新的iOS 11.x系统的设备仅占7.7%，有近半的iOS设备系统版本为10.3.3。其余设备停留在10.3.2到更早的7.x等不同的版本。这些旧版iOS系统的设备（高达92.3%）面临着文中列举的WiFi芯片高危漏洞带来的安全威胁。

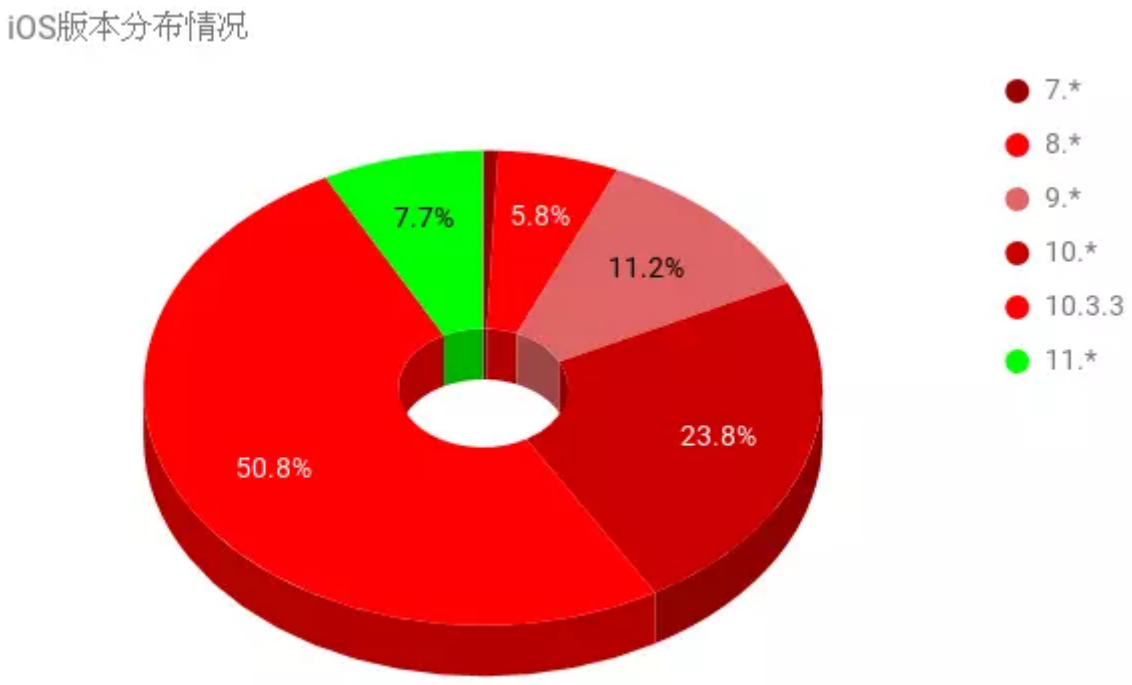


图4：国内iOS设备系统版本分布（2017年9月27日）

此外，我们统计了从2017年9月11日到2017年9月27日（iOS 11的升级推出于9月19日）iOS各个系统版本的升级情况。结果如图5所示。可以看到，从9月19日开始推送iOS 11.0升级至9月27日期间，iOS 11.x用户占比缓慢递增至7.7%，升级主要来源于iOS 10.3.3。整个统计期间iOS 7.x至10.x的用户占比基本不变，也就是说，仍然有近41.5%的用户选择停留在iOS 10.3.2及以下系统不升级。

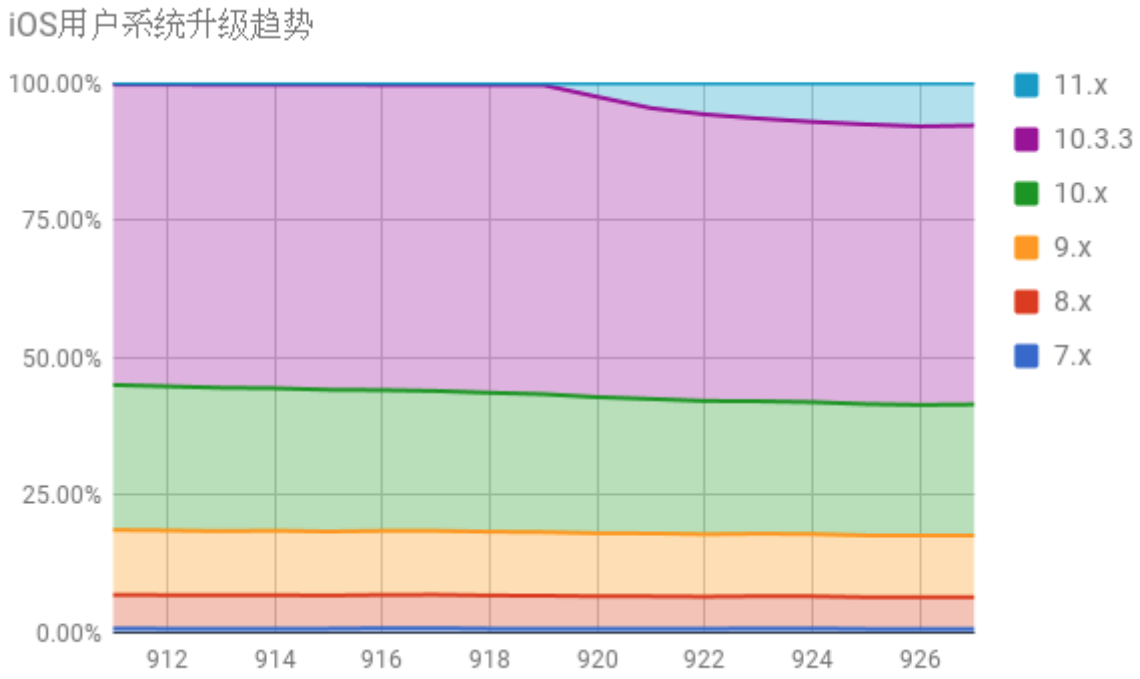


图5：国内iOS设备系统升级趋势（2017年9月11日-2017年9月27日）

## 5. 结语

本文分析了近期曝出的iOS设备WiFi芯片高危漏洞以及影响范围。文中提及的漏洞虽然已在iOS 11得到修复，但通过实际统计我们发现，绝大多数用户由于各种原因没有升级上来，仍面临严重的安全威胁。我们建议用户及时升级系统到最新版本，避免受到高危漏洞影响。同时，我们也呼吁手机以及硬件厂商采取更积极的防护技术，避免用户受到已知的高危漏洞威胁。

## 参考文献

[1]. https://support.apple.com/en-us/HT208112

[2]. https://bugs.chromium.org/p/project-zero/issues/detail?id=1289

[3]. https://bugs.chromium.org/p/project-zero/issues/detail?id=1291

[4]. https://bugs.chromium.org/p/project-zero/issues/detail?id=1302

[5].https://bugs.chromium.org/p/project-zero/issues/detail?id=1305

[6].https://bugs.chromium.org/p/project-zero/issues/detail?id=1312&can=1&q=CVE-2017-7108

[7].https://bugs.chromium.org/p/project-zero/issues/detail?id=1313&can=1&q=CVE-2017-7110

[8].https://bugs.chromium.org/p/projectzero/issues/detailid=1314&can=1&q=owner%3Alaginimaineb%40google.com