Igor Atsberger

PING OF DEATH 
ON FREEBSD 
(CVE-2018-17156)

Essay

Tartu 2019
Introduction

This essay gives overview of a Ping of death security vulnerability on FreeBSD system 11.0 and 11.1 versions that occurs due to incorrectly accounting for padding on 64-bit platforms, a buffer underwrite could occur when constructing an ICMP reply packet when using a non-standard value for the net.inet.icmp.quotelen sysctl (since versions from July 2018 works also with default value).

ICMP stands for Internet Control Message Protocol. The vulnerability could be revealed when replying to ICMP packet (for example replying to ping).

Description of vulnerability

From FreeBSD documentation:

"The icmp_error routine allocates either an mbuf or a cluster depending on the size of the data to be quoted in the ICMP reply, but the calculation failed to account for additional padding on 64-bit platforms when using a non-default sysctl value for net.inet.icmp.quotelen. For systems that set net.inet.icmp.quotelen to a non-default value, a buffer underwrite condition occurs."

From FreeBSD ICMP Manual:

"quotelen (integer) Number of bytes from original packet to quote in ICMP reply. This number is internally enforced to be at least 8 bytes (per RFC792) and at most the maximal space left in the ICMP reply mbuf."

A buffer underrun is a state when program tries to read from an empty buffer which leads to reading data outside the designated memory area, in other words program reads data of other program, which may lead to incorrect behaviour of the first program and corrupted data for the second program. The severity of consequences depends on certain programs, on memory allocation and the importance of the data.

The following is part of icmp_error() function in sys/netinet/ip_icmp.c commented with values of variables:

```c
oiphlen = oip->ip_hl << 2;
// oiphlen = 20
nlen = m_length(n, NULL);
```
\[ \text{nlen} = [137, 139] \]

\[ \text{icmpelen} = \max(8, \min(V_{\text{icmp_quotelen}}, \text{ntohs(oip->ip_len)} - \text{oiphlen})) ; \]

\[ \text{icmpelen} = \max(8, \min(548, [137, 139] - 20)) \]
\[ \text{icmpelen} = \max(8, \min(548, [117, 119])) \]
\[ \text{icmpelen} = [117, 119] \]
\[ \text{icmplen} = \min(\text{oiphlen} + \text{icmpelen}, \text{nlen}) ; \]

\[ \text{icmplen} = \min(20 + [117, 119], [137, 139]) \]
\[ \text{icmplen} = \min([137, 139], [137, 139]) \]
\[ \text{icmplen} = [137, 139] \]

if (MHLEN > sizeof(struct ip) + ICMP_MINLEN + icmplen)
    m = m_gethdr(M_NOWAIT, MT_DATA);

\[ \text{MHLEN} = 168 \text{ on 64-bit} \]
\[ \text{if} (168 > 20 + 8 + [137, 139]) \]
\[ \text{if} (168 > [165, 167]) \]
\[ \text{yes} \]
\[ \text{icmplen} = \min(\text{icmplen}, \text{M_TRAILINGSPACE}(m) - \text{sizeof(struct ip)} - ICMP_MINLEN) ; \]

\[ \text{icmplen} = \min([137, 139], \text{MHLEN} - 20 - 8) \]
\[ \text{icmplen} = \min([137, 139], 168 - 20 - 8) \]
\[ \text{icmplen} = \min([137, 139], 140) \]
\[ \text{icmplen} = [137, 139] \]

m_align(m, ICMP_MINLEN + icmplen);

\[ \text{m_align}(m, 8 + [137, 139]) \]
\[ \text{m_align}(m, [145, 147]) \]

// from the m_align() definition:
// adjust = M_SIZE(m) - len;
// m->m_data += adjust &~ (sizeof(long)-1);
// adjust = MHLEN - [145, 147]
// adjust = 168 - [145, 147]
// adjust = [21, 23]
// [21, 23] &~ (sizeof(long) - 1)
// [21, 23] &~ 7 (LP64)
// 16
// m->m_data += 16
m->m_data -= sizeof(struct ip);
...

nip = mtod(m, struct ip *);
bcopy((caddr_t)oip, (caddr_t)nip, sizeof(struct ip));

// 4-byte underwrite

As visible the size of reply is calculated incorrectly, therefore underwrite occurs.

Exploit

Luckily the problem is not that severe and as is stated in link #1 “underwrite hits a harmless area of the mbuf pkthdr” which is header of the buffer underwrite occurs in. However after that m_data will remain to point 4 bytes out of buffer. So calling this function from somewhere else may cause some damage. So exploiting is unlikely but still there is potential for doing so.

Fix

Workaround: set net.inet.icmp.quotelen to default value of 8 using sysctl(8): # sysctl net.inet.icmp.quotelen=8 (for older versions).
Workaround 2: set firewall and drop any ping packets at firewall.

Actual fix from FreeBSD’s GitHub (link 2):

As visible, the alignment is fixed and check added so that danger of underwrite would cause kernel panic through KASSERT function.
Used links

1) https://www.reddit.com/r/BSD/comments/9v6xwg/remotely_triggerable_icmp_buffer_underwrite_in/
2) https://github.com/freebsd/freebsd/commit/a9fb9d9a3082845e55798950d85fe93a61a8630a#diff-769daffb0659a28ad8a69b0f3551ab9f
7) https://www.freebsd.org/cgi/man.cgi?query=icmp&sektion=4