

Re: panic: sackhint rexmit == 0

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/current/2005-08/1303.html>

From: Noritoshi Demizu (*demizu_at_dd.iij4u.or.jp*)

Date: 08/30/05

Date: Tue, 30 Aug 2005 12:06:04 +0900 (JST)

To: Paul Saab <ps@freebsd.org>

> > #23 0xc063ae06 in tcp_input (m=0xc1e56400, off0=20)
> > at /usr/home/build/src/sys/netinet/tcp_input.c:1915
> > 1915 KASSERT(tp->sackhint.

> *Should already be fixed in -current.*

The change of tcp_input.c rev 1.283 does not *fix* the problem.
It is just a workaround of the problem.

I believe the real problem is that tp->snd_cwnd, tp->snd_ssthresh, tp->snd_recover, and tp->snd_nxt are falsely recovered by an algorithm using tp->t_badrxtwin in the following scenario.

1. An Retransmission Timeout occurs. Now, t_rxtshift is 1.
(snd_cwnd, snd_ssthresh, snd_recover are saved in tcp_timer_rexmt().)
2. Before t_rxtshift is reset to zero, Fast Retransmit is triggered and TCP enters Fast Recovery. (Note: t_rxtshift is reset to zero by tcp_xmit_timer() when a new RTT measurement is taken. If my memory serves correctly, this behavior is specified in RFC2988).
3. A partial ACK or a full ACK is received before "ticks" reaches at tp->t_badrxtwin.

In this case, lines 2047-2056 in tcp_input.c 1.283 recovers snd_cwnd, snd_ssthresh, snd_recover and snd_nxt.
(Note: t_rxtshift has not been reset to zero. It will be reset at line 2074 or 2076 of tcp_input.c 1.283.)

I believe those variables must not be recovered in this case.
Since snd_recover is recovered, snd_recover becomes smaller than the actual value. Hence, the condition at line 2134 of tcp_input.c 1.283 falsely becomes true, and TCP falsely exits Fast Recovery.
It breaks internal states of TCP SACK.

In tcp_input.c 1.282 or before, the possible corruption was detected

freebsd-current: Re: panic: sackhint rexmit == 0

by the lines removed in the change of tcp_input.c 1.283. Since the check was not so significant, the check was removed as a workaround.

I believe the following change fixes the problem by avoiding step 3 above. Introducing a new flag indicating whether t_badrxtwin is valid would be a better solution because a TCP connection may live longer than 2^{32} ticks of time.

```
<<Quoted from tcp_input.c rev 1.283>>  
1915: ENTER_FASTRECOVERY(tp);  
1916: tp->snd_recover = tp->snd_max;  
1917: callout_stop(tp->tt_rexmt);  
1918: tp->t_rtttime = 0;  
+ tp->t_badrxtwin = ticks;  
1919: if (tp->sack_enable) {
```

Regards,
Noritoshi Demizu

freebsd-current@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-current>

To unsubscribe, send any mail to "freebsd-current-unsubscribe@freebsd.org"