ping: Fix signed 64-bit integer overflow in RTT calculation #585

New issue

Open **pevik** wants to merge 1 commit into `iputils:master` from `pevik:CVE-2025-47268`

pevik commented 16 hours ago Contributor

Crafted ICMP Echo Reply packet can cause signed integer overflow in

1. triptime calculation:
   triptime = tv->tv_sec * 1000000 + tv->tv_usec;

2. tsum2 increment which uses triptime
   rts->tsum2 += (double)((long long)triptime * (long long)triptime);

3. final tmvar:
   tmvar = (rts->tsum2 / total) - (tmavg * tmavg)

```
$ export CFLAGS="-O1 -g -fsanitize=address,undefi
-fno-omit-frame-pointer"
$ export LDFLAGS="-fsanitize=address,undefined -fno-
omit-frame-pointer"
$ meson setup .. -Db_sanitize=address,undefined
$ ninja
$ ./ping/ping -c2 127.0.0.1

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64
time=0.061 ms
../ping/ping_common.c:757:25: runtime error: signed
integer overflow: -2513732689199106 * 1000000 cannot
be represented in type 'long int'
../ping/ping_common.c:757:12: runtime error: signed
integer overflow: -4975495174606980224 +
-6510615555425289427 cannot be represented in type
'long int'
../ping/ping_common.c:769:47: runtime error: signed
integer overflow: 6960633343677281965 *
6960633343677281965 cannot be represented in type
'long int'
```

Reviewers

nmeyerhans
metan-ucw
Zephkek

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

Signed 64-bit integer overflow in RTT ...

4 participants

```
24 bytes from 127.0.0.1: icmp_seq=1 ttl=64
(truncated)
./ping/ping: Warning: time of day goes back
(-7256972569576721377us), taking countermeasures
./ping/ping: Warning: time of day goes back
(-7256972569576721232us), taking countermeasures
24 bytes from 127.0.0.1: icmp_seq=1 ttl=64
(truncated)
../ping/ping_common.c:265:16: runtime error: signed
integer overflow: 6960633343677281965 * 2 cannot be
represented in type 'long int'
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64
time=0.565 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0%
packet loss, time 1002ms
../ping/ping_common.c:940:42: runtime error: signed
integer overflow: 1740158335919320832 *
1740158335919320832 cannot be represented in type
'long int'
rtt min/avg/max/mdev =
0.000/1740158335919320.832/6960633343677281.965/-162351
ms
```

To fix the overflow check allowed ranges of struct timeval members:

- tv_sec <-LONG_MAX/1000000, LONG_MAX/1000000>
- tv_usec <0, 999999>

Fix includes 2 new error messages (needs translation).

After fix:

```
$ ./ping/ping -c2 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64
time=0.059 ms
./ping/ping: Warning: overflow tv_usec
-6510615555425457380 us
./ping/ping: Warning: invalid tv_sec
-1789369274859522 s
24 bytes from 127.0.0.1: icmp_seq=1 ttl=64
(truncated)
./ping/ping: Warning: overflow tv_usec
-6510615555425413387 us
./ping/ping: Warning: invalid tv_sec
-2006106209517570 s
24 bytes from 127.0.0.1: icmp_seq=1 ttl=64
(truncated)
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64
time=0.118 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0%
```

```
packet loss, time 1002ms
rtt min/avg/max/mdev = 0.000/0.044/0.118/0.048 ms
```

Fixes: #584
Fixes: CVE-2025-472
Link: https://github.com/Zephkek/ping-rtt-overflow/
Co-developed-by: Cyril Hrubis chrubis@suse.cz
Reported-by: Mohamed Maatallah
hotelsmaatallahrecemail@gmail.com

pevik mentioned this pull request 16 hours ago

Signed 64-bit integer overflow in RTT calculation #584 Open

pevik requested a review from **a team** 16 hours ago

nmeyerhans reviewed 15 hours ago View reviewed changes

ping/ping_common.c Outdated

```
+                 }
+
+                 /* 1000001 = 1000000 tv
+                 if (tv->tv_sec > LONG_M
```

nmeyerhans 15 hours ago Contributor

It might be nice to give a name to `LONG_MAX/1000001`, maybe with `#define`?

👍 1

Zephkek 15 hours ago • edited

Something like this would be nice:

```
#define USEC_PER_SEC 1000000
#define USEC_MAX (USEC_PER_SEC - 1)
#define SEC_SAFE_MAX (LONG_MAX / (USEC_PER_SE
```

These replace magic numbers and create a clear boundary for detecting integer overflow when converting time units.

pevik 15 hours ago Contributor Author

Although the other 2 definitions make sense, I would prefer to postpone adding them later after this is fixed (it's an unrelated cleanup - `1000000` should be used on more places not just here).

pevik force-pushed the `CVE-2025-47268` branch 2 times, most recently from **`d17b6d0`** to **`23db9b0`** 15 hours ago

Compare

pevik commented 9 hours ago Contributor Author

Please, when you're finish with your review, add your `Reviewed-by:` or `Acked-by:` tag.

pevik requested a review from **a team** 8 hours ago

metan-ucw reviewed 8 hours ago

View reviewed changes

ping/ping_common.c

```
765 +                        tv->tv_usec = (
766 +                }
767 +
768 +                if (tv->tv_sec > TV_SEC
```

metan-ucw 8 hours ago Contributor

Isn't tv->tv_sec < 0 invalid anyway? That would mean that the packet traveled back in time.

pevik 8 hours ago • edited Contributor Author

It makes sense, but IMHO this is handled later this part:

iputils/ping/ping_common.c
Lines 758 to 766 in 3bb2d73

```
758     if (triptime < 0) {
759             error(0, 0, _("Warning: time of day goes back (%ldus), taking countermeasures"), triptime);
760             triptime = 0;
761             if (!rts->opt_latency) {
762                     gettimeofday(tv, NULL);
```

```
                    rts->opt_latency = 1;
                    goto restamp;
```

I also wondered if I should move it to handle it via `tv->tv_sec < 0` as we now sanitize `tv->tv_usec` (I guess we should keep `time of day goes back` warning message for it). But how about `if (!rts->opt_latency) { ... }` part? Is it relevant for crafted RTT values as well?

metan-ucw 8 hours ago Contributor

At the start of the `gather_statistics()` we do `tv_sub()` where we calculate the difference between the time we send the packet and the time we received a reply. We have no way knowing if we got negative value because of wall clock change or because of a crafted value.

And in the case of the crafted value the problem is even worse, I guess that if we send a timestamp that is ahead by hours ping will get stuck in the loop, trying to restamp it for hours consuming 100% of CPU time. It would make more sense to discard such sample from the statistics.

So I would do:

- Remove the restamp goto
- Check for negative value right in the tv_sec and set triptime to 0 if it was negative
- Skip the part where we add to the rts->tsum if triptime == 0

metan-ucw 7 hours ago Contributor

Ah I was blind, we actually use rts->opt_latency to guard against infinite loop. However the restamping is still questionable, we are not getting a good sample by pretending it arrived a tiny bit later.

Also idea for a future, we should switch to `CLOCK_MONOTONIC` timer that is not going to go backwards unlike the wall clock.

Zephkek 7 hours ago • edited

Using `CLOCK_MONOTONIC` for send times seems practical initially, potentially obtaining a `timespec` but converting to `timeval` for the ICMP payload. This mainly improves the reliability of RTTs at the current microsecond precision by avoiding wall-clock issues.

For receive times, kernel monotonic `timespec` timestamps (via `SO_TIMESTAMPNS` / `SO_TIMESTAMPING`) would be ideal, with user-space `clock_gettime(CLOCK_MONOTONIC)` post- `recvmsg` as a fallback.

This isolates RTT from wall-clock changes, yielding more reliable results.

Load more...

metan-ucw 5 hours ago Contributor

No need for the else branch, we can do:

```
 if (tv->tv_sec > TV_SEC_MAX_VAL) {

 }

if (tv->tv_sec < 0) {

}
```

Apart from that it does sound like a plan to me. Let's limit the changes for this particular fix and then do a bigger cleanup once this is dealt with.

pevik 4 hours ago Contributor Author

The only thing is that keeping also negative separate

```
if (tv->tv_sec > TV_SEC_MAX_VAL || tv->
    /* underflow or overflow => likely
} else if (triptime < 0) {
    /*
     * Negative value but small enough to be
     * => I would keep the warning about time
     */
}
```

e.g. to have final code

```
restamp:
                tvsub(tv, &tmp_tv);

                if (tv->tv_usec >= 1000000) {
                        error(0, 0, _("Warnir
                        tv->tv_usec = 999999;
                }

                if (tv->tv_usec < 0) {
                        error(0, 0, _("Warnir
                        tv->tv_usec = 0;
                }

                if (tv->tv_sec > TV_SEC_MAX_\
                        error(0, 0, _("Warnir
                        triptime = 0;
                } else if (tv->tv_sec < 0) {
                        error(0, 0, _("Warnir
                        triptime = 0;
                        if (!rts->opt_latency
                                gettimeofday(
                                rts->opt_late
                                goto restamp;
                        }
                } else {
                        triptime = tv->tv_sec
                }

                if (!csfailed) {
...
```

Or you meant something different?

metan-ucw 4 hours ago • edited Contributor

You left in the `|| tv->tv_sec < -TV_SEC_MAX_VAL` that shouldn't be needed because we without that we would end up in the `if (tv->tv_sec < 0)` branch. Otherwise it looks good.

pevik 4 hours ago Contributor Author

That was deliberate: e.g. first check for `long int` overflow (outside of range: `<-9223362813491, 9223362813491>`), then check for smaller negative value in range `<-9223362813490, 0)` which could be also time back. I think that `<-9223362813490, 0)` is very long interval, but do you really consider not checking for invalid negative range useful? Because the crafting script sets: `tv->tv_sec: -2601281302560770, tv->tv_usec: -6510615555425262901` . Therefore the output has:

```
./ping/ping: Warning: overflow tv_usec
-6510615555425218641 us
./ping/ping: Warning: invalid tv_sec
-2821724793995266 s
```

(Maybe `tv->tv_usec` could have error message just `invalid tv_usec` - not specify overflow/underflow. Or, if kept, then `tv->tv_sec` should also specify overflow/underflow).

metan-ucw 2 hours ago • edited Contributor

If we are not doing to do the multiplication in the case that `tv_sec < 0` then there is no point in checking for underflow and no point in treating some negative numbers differently. At least that is my reasoning why there is no need to treat a subset of negative numbers differently.

👍 1

`ping: Fix signed 64-bit integer overflow in RTT calculation` Verified ✓ `4c799c7`

pevik force-pushed the `CVE-2025-47268` branch from **`23db9b0`** to **`4c799c7`** 5 hours ago Compare

pevik commented 5 hours ago Contributor Author

Branch rebased.

nmeyerhans commented 31 minutes ago Contributor

> Please, when you're finish with your review, add your `Reviewed-by:` or `Acked-by:` tag.

Acked in `c03bb27` in my fork