author	Daniel Borkmann <daniel@iogearbox.net>	2025-04-23 15:36:00 +0200
committer	Jakub Kicinski <kuba@kernel.org>	2025-04-25 17:24:07 -0700
commit	4c2227656d9003f4d77afc76f34dd81b95e4c2c4 (patch)	
tree	cbfce489a43cb856b30a6d41af56e62ff0b3e3c8	
parent	49ba1ca2e0cc6d2eb0667172f1144c8b85907971 (diff)	
download	linux-4c2227656d9003f4d77afc76f34dd81b95e4c2c4.tar.gz	

vmxnet3: Fix malformed packet sizing in vmxnet3_process_xdp

vmxnet3 driver's XDP handling is buggy for packet sizes using ring0 (that
is, packet sizes between 128 - 3k bytes).

We noticed MTU-related connectivity issues with Cilium's service load-
balancing in case of vmxnet3 as NIC underneath. A simple curl to a HTTP
backend service where the XDP LB was doing IPIP encap led to overly large
packet sizes but only for *some* of the packets (e.g. HTTP GET request)
while others (e.g. the prior TCP 3WHS) looked completely fine on the wire.

In fact, the pcap recording on the backend node actually revealed that the
node with the XDP LB was leaking uninitialized kernel data onto the wire
for the affected packets, for example, while the packets should have been
152 bytes their actual size was 1482 bytes, so the remainder after 152 bytes
was padded with whatever other data was in that page at the time (e.g. we
saw user/payload data from prior processed packets).

We only noticed this through an MTU issue, e.g. when the XDP LB node and
the backend node both had the same MTU (e.g. 1500) then the curl request
got dropped on the backend node's NIC given the packet was too large even
though the IPIP-encapped packet normally would never even come close to
the MTU limit. Lowering the MTU on the XDP LB (e.g. 1480) allowed to let
the curl request succeed (which also indicates that the kernel ignored the
padding, and thus the issue wasn't very user-visible).

Commit e127ce7699c1 ("vmxnet3: Fix missing reserved tailroom") was too eager
to also switch xdp_prepare_buff() from rcd->len to rbi->len. It really needs
to stick to rcd->len which is the actual packet length from the descriptor.
The latter we also feed into vmxnet3_process_xdp_small(), by the way, and
it indicates the correct length needed to initialize the xdp->{data,data_end}
parts. For e127ce7699c1 ("vmxnet3: Fix missing reserved tailroom") the
relevant part was adapting xdp_init_buff() to address the warning given the
xdp_data_hard_end() depends on xdp->frame_sz. With that fixed, traffic on
the wire looks good again.

Fixes: e127ce7699c1 ("vmxnet3: Fix missing reserved tailroom")
Signed-off-by: Daniel Borkmann <daniel@iogearbox.net>
Tested-by: Andrew Sauber <andrew.sauber@isovalent.com>
Cc: Anton Protopopov <aspsk@isovalent.com>
Cc: William Tu <witu@nvidia.com>
Cc: Martin Zaharinov <micron10@gmail.com>
Cc: Ronak Doshi <ronak.doshi@broadcom.com>

Reviewed-by: Simon Horman <horms@kernel.org>
Link: https://patch.msgid.link/20250423133600.176689-1-daniel@iogearbox.net
Signed-off-by: Jakub Kicinski <kuba@kernel.org>

Diffstat

-rw-r--r-- drivers/net/vmxnet3/vmxnet3_xdp.c 2

1 files changed, 1 insertions, 1 deletions

```
diff --git a/drivers/net/vmxnet3/vmxnet3_xdp.c b/drivers/net/vmxnet3/vmxnet3_xdp.c
index 616ecc38d1726c..5f470499e60024 100644
--- a/drivers/net/vmxnet3/vmxnet3_xdp.c
+++ b/drivers/net/vmxnet3/vmxnet3_xdp.c
@@ -397,7 +397,7 @@ vmxnet3_process_xdp(struct vmxnet3_adapter *adapter,

        xdp_init_buff(&xdp, PAGE_SIZE, &rq->xdp_rxq);
        xdp_prepare_buff(&xdp, page_address(page), rq->page_pool->p.offset,
-                        rbi->len, false);
+                        rcd->len, false);
        xdp_buff_clear_frags_flag(&xdp);

        xdp_prog = rcu_dereference(rq->adapter->xdp_bpf_prog);
```