author	shamir rabinovitch <shamir.rabinovitch@oracle.com> 2018-12-16 09:01:08 +0200
committer	David S. Miller <davem@davemloft.net> 2018-12-19 10:27:58 -0800
commit	ea010070d0a7497253d5a6f919f6dd107450b31a (patch)
tree	4bd1b33edd72bdd73d39e7f8160a73730704fa56
parent	c6f4075e2f14a91f2180c98bc7715946f791cbe6 (diff)
download	linux-ea010070d0a7497253d5a6f919f6dd107450b31a.tar.gz

diff options

context: 3
space: include
mode: unified

net/rds: fix warn in rds_message_alloc_sgs

```
redundant copy_from_user in rds_sendmsg system call expose rds
to issue where rds_rdma_extra_size walk the rds iovec and and
calculate the number pf pages (sgs) it need to add to the tail of
rds message and later rds_cmsg_rdma_args copy the rds iovec again
and re calculate the same number and get different result causing
WARN_ON in rds_message_alloc_sgs.

fix this by doing the copy_from_user only once per rds_sendmsg
system call.

When issue occur the below dump is seen:

WARNING: CPU: 0 PID: 19789 at net/rds/message.c:316 rds_message_alloc_sgs+0x10c/0x160 net/rds/message.c:316
Kernel panic - not syncing: panic_on_warn set ...
CPU: 0 PID: 19789 Comm: syz-executor827 Not tainted 4.19.0-next-20181030+ #101
Hardware name: Google Google Compute Engine/Google Compute Engine, BIOS Google 01/01/2011
Call Trace:
 __dump_stack lib/dump_stack.c:77 [inline]
 dump_stack+0x244/0x39d lib/dump_stack.c:113
 panic+0x2ad/0x55c kernel/panic.c:188
 __warn.cold.8+0x20/0x45 kernel/panic.c:540
 report_bug+0x254/0x2d0 lib/bug.c:186
 fixup_bug arch/x86/kernel/traps.c:178 [inline]
 do_error_trap+0x11b/0x200 arch/x86/kernel/traps.c:271
 do_invalid_op+0x36/0x40 arch/x86/kernel/traps.c:290
 invalid_op+0x14/0x20 arch/x86/entry/entry_64.S:969
RIP: 0010:rds_message_alloc_sgs+0x10c/0x160 net/rds/message.c:316
Code: c0 74 04 3c 03 7e 6c 44 01 ab 78 01 00 00 e8 2b 9e 35 fa 4c 89 e0 48 83 c4 08 5b 41 5c 41 5d 41 5e 41 5f 5d c3 e8 14 9e 35 fa <0f
RSP: 0018:ffff8801c51b7460 EFLAGS: 00010293
RAX: ffff8801bc412080 RBX: ffff8801d7bf4040 RCX: ffffffff8749c9e6
RDX: 0000000000000000 RSI: ffffffff8749ca5c RDI: 0000000000000004
RBP: ffff8801c51b7490 R08: ffff8801bc412080 R09: ffffed003b5c5b67
R10: ffffed003b5c5b67 R11: ffff8801dae2db3b R12: 0000000000000000
R13: 000000000007165c R14: 000000000007165c R15: 0000000000000005
 rds_cmsg_rdma_args+0x82d/0x1510 net/rds/rdma.c:623
 rds_cmsg_send net/rds/send.c:971 [inline]
 rds_sendmsg+0x19a2/0x3180 net/rds/send.c:1273
 sock_sendmsg_nosec net/socket.c:622 [inline]
 sock_sendmsg+0xd5/0x120 net/socket.c:632
 ___sys_sendmsg+0x7fd/0x930 net/socket.c:2117
 __sys_sendmsg+0x11d/0x280 net/socket.c:2155
 __do_sys_sendmsg net/socket.c:2164 [inline]
 __se_sys_sendmsg net/socket.c:2162 [inline]
 __x64_sys_sendmsg+0x78/0xb0 net/socket.c:2162
 do_syscall_64+0x1b9/0x820 arch/x86/entry/common.c:290
 entry_SYSCALL_64_after_hwframe+0x49/0xbe
RIP: 0033:0x44a859
Code: e8 dc e6 ff ff 48 83 c4 18 c3 0f 1f 80 00 00 00 00 48 89 f8 48 89 f7 48 89 d6 48 89 ca 4d 89 c2 4d 89 c8 4c 8b 4c 24 08 0f 05 <48
RSP: 002b:00007f1d4710ada8 EFLAGS: 00000297 ORIG_RAX: 000000000000002e
RAX: ffffffffffffffda RBX: 00000000006dcc28 RCX: 000000000044a859
RDX: 0000000000000000 RSI: 0000000020001600 RDI: 0000000000000003
RBP: 00000000006dcc20 R08: 0000000000000000 R09: 0000000000000000
R10: 0000000000000000 R11: 0000000000000297 R12: 00000000006dcc2c
R13: 646e732f7665642f R14: 00007f1d4710b9c0 R15: 00000000006dcd2c
Kernel Offset: disabled
Rebooting in 86400 seconds..

Reported-by: syzbot+26de17458aeda9d305d8@syzkaller.appspotmail.com
Acked-by: Santosh Shilimkar <santosh.shilimkar@oracle.com>
Signed-off-by: shamir rabinovitch <shamir.rabinovitch@oracle.com>
Signed-off-by: David S. Miller <davem@davemloft.net>
```

Diffstat

-rw-r--r--	net/rds/rdma.c	63
-rw-r--r--	net/rds/rds.h	20
-rw-r--r--	net/rds/send.c	50

3 files changed, 91 insertions, 42 deletions

```
diff --git a/net/rds/rdma.c b/net/rds/rdma.c
index 98237feb607ac..e1965d9cbcf82 100644
--- a/net/rds/rdma.c
+++ b/net/rds/rdma.c
@@ -517,9 +517,10 @@ static int rds_rdma_pages(struct rds_iovec iov[], int nr_iovecs)
        return tot_pages;
```

```
 }

-int rds_rdma_extra_size(struct rds_rdma_args *args)
+int rds_rdma_extra_size(struct rds_rdma_args *args,
+                        struct rds_iov_vector *iov)
 {
-       struct rds_iovec vec;
+       struct rds_iovec *vec;
        struct rds_iovec __user *local_vec;
        int tot_pages = 0;
        unsigned int nr_pages;
@@ -530,13 +531,23 @@ int rds_rdma_extra_size(struct rds_rdma_args *args)
        if (args->nr_local == 0)
                return -EINVAL;

+       iov->iov = kcalloc(args->nr_local,
+                          sizeof(struct rds_iovec),
+                          GFP_KERNEL);
+       if (!iov->iov)
+               return -ENOMEM;
+
+       vec = &iov->iov[0];
+
+       if (copy_from_user(vec, local_vec, args->nr_local *
+                          sizeof(struct rds_iovec)))
+               return -EFAULT;
+       iov->len = args->nr_local;
+
        /* figure out the number of pages in the vector */
-       for (i = 0; i < args->nr_local; i++) {
-               if (copy_from_user(&vec, &local_vec[i],
-                                  sizeof(struct rds_iovec)))
-                       return -EFAULT;
+       for (i = 0; i < args->nr_local; i++, vec++) {

-               nr_pages = rds_pages_in_vec(&vec);
+               nr_pages = rds_pages_in_vec(vec);
                if (nr_pages == 0)
                        return -EINVAL;

@@ -558,15 +569,15 @@ int rds_rdma_extra_size(struct rds_rdma_args *args)
  * Extract all arguments and set up the rdma_op
  */
 int rds_cmsg_rdma_args(struct rds_sock *rs, struct rds_message *rm,
-                       struct cmsghdr *cmsg)
+                      struct cmsghdr *cmsg,
+                      struct rds_iov_vector *vec)
 {
        struct rds_rdma_args *args;
        struct rm_rdma_op *op = &rm->rdma;
        int nr_pages;
        unsigned int nr_bytes;
        struct page **pages = NULL;
-       struct rds_iovec iovstack[UIO_FASTIOV], *iovs = iovstack;
-       int iov_size;
+       struct rds_iovec *iovs;
        unsigned int i, j;
        int ret = 0;

@@ -586,31 +597,23 @@ int rds_cmsg_rdma_args(struct rds_sock *rs, struct rds_message *rm,
                goto out_ret;
        }

-       /* Check whether to allocate the iovec area */
-       iov_size = args->nr_local * sizeof(struct rds_iovec);
-       if (args->nr_local > UIO_FASTIOV) {
-               iovs = sock_kmalloc(rds_rs_to_sk(rs), iov_size, GFP_KERNEL);
-               if (!iovs) {
-                       ret = -ENOMEM;
-                       goto out_ret;
-               }
+       if (vec->len != args->nr_local) {
+               ret = -EINVAL;
+               goto out_ret;
        }

-       if (copy_from_user(iovs, (struct rds_iovec __user *)(unsigned long) args->local_vec_addr, iov_size)) {
-               ret = -EFAULT;
-               goto out;
-       }
+       iovs = vec->iov;

        nr_pages = rds_rdma_pages(iovs, args->nr_local);
        if (nr_pages < 0) {
                ret = -EINVAL;
-               goto out;
+               goto out_ret;
        }

        pages = kcalloc(nr_pages, sizeof(struct page *), GFP_KERNEL);
        if (!pages) {
                ret = -ENOMEM;
-               goto out;
+               goto out_ret;
        }

        op->op_write = !!(args->flags & RDS_RDMA_READWRITE);
@@ -623,7 +626,7 @@ int rds_cmsg_rdma_args(struct rds_sock *rs, struct rds_message *rm,
```

```
        op->op_sg = rds_message_alloc_sgs(rm, nr_pages);
        if (!op->op_sg) {
                ret = -ENOMEM;
-               goto out;
+               goto out_pages;
        }

        if (op->op_notify || op->op_recverr) {
@@ -635,7 +638,7 @@ int rds_cmsg_rdma_args(struct rds_sock *rs, struct rds_message *rm,
                op->op_notifier = kmalloc(sizeof(struct rds_notifier), GFP_KERNEL);
                if (!op->op_notifier) {
                        ret = -ENOMEM;
-                       goto out;
+                       goto out_pages;
                }
                op->op_notifier->n_user_token = args->user_token;
                op->op_notifier->n_status = RDS_RDMA_SUCCESS;
@@ -681,7 +684,7 @@ int rds_cmsg_rdma_args(struct rds_sock *rs, struct rds_message *rm,
                 */
                ret = rds_pin_pages(iov->addr, nr, pages, !op->op_write);
                if (ret < 0)
-                       goto out;
+                       goto out_pages;
                else
                        ret = 0;

@@ -714,13 +717,11 @@ int rds_cmsg_rdma_args(struct rds_sock *rs, struct rds_message *rm,
                                  nr_bytes,
                                  (unsigned int) args->remote_vec.bytes);
                ret = -EINVAL;
-               goto out;
+               goto out_pages;
        }
        op->op_bytes = nr_bytes;

-out:
-       if (iovs != iovstack)
-               sock_kfree_s(rds_rs_to_sk(rs), iovs, iov_size);
+out_pages:
        kfree(pages);
 out_ret:
        if (ret)

diff --git a/net/rds/rds.h b/net/rds/rds.h
index 6bfaf05b63b21..4d2523100093b 100644
--- a/net/rds/rds.h
+++ b/net/rds/rds.h
@@ -386,6 +386,18 @@ static inline void rds_message_zcopy_queue_init(struct rds_msg_zcopy_queue *q)
        INIT_LIST_HEAD(&q->zcookie_head);
 }

+struct rds_iov_vector {
+       struct rds_iovec *iov;
+       int               len;
+};
+
+struct rds_iov_vector_arr {
+       struct rds_iov_vector *vec;
+       int                    len;
+       int                    indx;
+       int                    incr;
+};
+
 struct rds_message {
        refcount_t              m_refcount;
        struct list_head        m_sock_item;
@@ -904,13 +916,13 @@ int rds_get_mr(struct rds_sock *rs, char __user *optval, int optlen);
 int rds_get_mr_for_dest(struct rds_sock *rs, char __user *optval, int optlen);
 int rds_free_mr(struct rds_sock *rs, char __user *optval, int optlen);
 void rds_rdma_drop_keys(struct rds_sock *rs);
-int rds_rdma_extra_size(struct rds_rdma_args *args);
-int rds_cmsg_rdma_args(struct rds_sock *rs, struct rds_message *rm,
-                       struct cmsghdr *cmsg);
+int rds_rdma_extra_size(struct rds_rdma_args *args,
+                       struct rds_iov_vector *iov);
 int rds_cmsg_rdma_dest(struct rds_sock *rs, struct rds_message *rm,
                        struct cmsghdr *cmsg);
 int rds_cmsg_rdma_args(struct rds_sock *rs, struct rds_message *rm,
-                       struct cmsghdr *cmsg);
+                       struct cmsghdr *cmsg,
+                       struct rds_iov_vector *vec);
 int rds_cmsg_rdma_map(struct rds_sock *rs, struct rds_message *rm,
                       struct cmsghdr *cmsg);
 void rds_rdma_free_op(struct rm_rdma_op *ro);

diff --git a/net/rds/send.c b/net/rds/send.c
index fe785ee819ddb..ec2267cbf85f0 100644
--- a/net/rds/send.c
+++ b/net/rds/send.c
@@ -876,13 +876,15 @@ out:
  * rds_message is getting to be quite complicated, and we'd like to allocate
  * it all in one go. This figures out how big it needs to be up front.
  */
-static int rds_rm_size(struct msghdr *msg, int num_sgs)
+static int rds_rm_size(struct msghdr *msg, int num_sgs,
+                      struct rds_iov_vector_arr *vct)
 {
        struct cmsghdr *cmsg;
```

```
        int size = 0;
        int cmsg_groups = 0;
        int retval;
        bool zcopy_cookie = false;
+       struct rds_iov_vector *iov, *tmp_iov;

        for_each_cmsghdr(cmsg, msg) {
                if (!CMSG_OK(msg, cmsg))
@@ -893,8 +895,24 @@ static int rds_rm_size(struct msghdr *msg, int num_sgs)

                switch (cmsg->cmsg_type) {
                case RDS_CMSG_RDMA_ARGS:
+                       if (vct->indx >= vct->len) {
+                               vct->len += vct->incr;
+                               tmp_iov =
+                                       krealloc(vct->vec,
+                                                vct->len *
+                                                sizeof(struct rds_iov_vector),
+                                                GFP_KERNEL);
+                               if (!tmp_iov) {
+                                       vct->len -= vct->incr;
+                                       return -ENOMEM;
+                               }
+                               vct->vec = tmp_iov;
+                       }
+                       iov = &vct->vec[vct->indx];
+                       memset(iov, 0, sizeof(struct rds_iov_vector));
+                       vct->indx++;
                        cmsg_groups |= 1;
-                       retval = rds_rdma_extra_size(CMSG_DATA(cmsg));
+                       retval = rds_rdma_extra_size(CMSG_DATA(cmsg), iov);
                        if (retval < 0)
                                return retval;
                        size += retval;
@@ -951,10 +969,11 @@ static int rds_cmsg_zcopy(struct rds_sock *rs, struct rds_message *rm,
 }

 static int rds_cmsg_send(struct rds_sock *rs, struct rds_message *rm,
-                        struct msghdr *msg, int *allocated_mr)
+                        struct msghdr *msg, int *allocated_mr,
+                        struct rds_iov_vector_arr *vct)
 {
        struct cmsghdr *cmsg;
-       int ret = 0;
+       int ret = 0, ind = 0;

        for_each_cmsghdr(cmsg, msg) {
                if (!CMSG_OK(msg, cmsg))
@@ -968,7 +987,10 @@ static int rds_cmsg_send(struct rds_sock *rs, struct rds_message *rm,
                 */
                switch (cmsg->cmsg_type) {
                case RDS_CMSG_RDMA_ARGS:
-                       ret = rds_cmsg_rdma_args(rs, rm, cmsg);
+                       if (ind >= vct->indx)
+                               return -ENOMEM;
+                       ret = rds_cmsg_rdma_args(rs, rm, cmsg, &vct->vec[ind]);
+                       ind++;
                        break;

                case RDS_CMSG_RDMA_DEST:
@@ -1084,6 +1106,11 @@ int rds_sendmsg(struct socket *sock, struct msghdr *msg, size_t payload_len)
                      sock_flag(rds_rs_to_sk(rs), SOCK_ZEROCOPY));
        int num_sgs = ceil(payload_len, PAGE_SIZE);
        int namelen;
+       struct rds_iov_vector_arr vct = {0};
+       int ind;
+
+       /* expect 1 RDMA CMSG per rds_sendmsg. can still grow if more needed. */
+       vct.incr = 1;

        /* Mirror Linux UDP mirror of BSD error message compatibility */
        /* XXX: Perhaps MSG_MORE someday */
@@ -1220,7 +1247,7 @@ int rds_sendmsg(struct socket *sock, struct msghdr *msg, size_t payload_len)
                num_sgs = iov_iter_npages(&msg->msg_iter, INT_MAX);
        }
        /* size of rm including all sgs */
-       ret = rds_rm_size(msg, num_sgs);
+       ret = rds_rm_size(msg, num_sgs, &vct);
        if (ret < 0)
                goto out;

@@ -1270,7 +1297,7 @@ int rds_sendmsg(struct socket *sock, struct msghdr *msg, size_t payload_len)
        rm->m_conn_path = cpath;

        /* Parse any control messages the user may have included. */
-       ret = rds_cmsg_send(rs, rm, msg, &allocated_mr);
+       ret = rds_cmsg_send(rs, rm, msg, &allocated_mr, &vct);
        if (ret) {
                /* Trigger connection so that its ready for the next retry */
                if (ret ==  -EAGAIN)
@@ -1348,9 +1375,18 @@ int rds_sendmsg(struct socket *sock, struct msghdr *msg, size_t payload_len)
        if (ret)
                goto out;
        rds_message_put(rm);
+
+       for (ind = 0; ind < vct.indx; ind++)
+               kfree(vct.vec[ind].iov);
+       kfree(vct.vec);
```

```
+
        return payload_len;

 out:
+       for (ind = 0; ind < vct.indx; ind++)
+               kfree(vct.vec[ind].iov);
+       kfree(vct.vec);
+
        /* If the user included a RDMA_MAP cmsg, we allocated a MR on the fly.
         * If the sendmsg goes through, we keep the MR. If it fails with EAGAIN
         * or in any other way, we need to destroy the MR again */
```