

---

Subject: [PATCH COMMIT] diff-merge-2.6.16.15-20060510

Posted by [xemul](#) on Wed, 10 May 2006 10:28:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Added to 026test012

Patch from OpenVZ team <[devel@openvz.org](mailto:devel@openvz.org)>

Merged 2.6.16.15 from /linux/kernel/git/stable/linux-2.6.16.y

From: OpenVZ team <[devel@openvz.org](mailto:devel@openvz.org)>

Date: Wed, 10 May 2006 10:24:44 +0000 (+0400)

Subject: Merged 2.6.16.15 from /linux/kernel/git/stable/linux-2.6.16.y

X-Git-Url:

<http://10.11.12.17/cgi-bin/gitweb.cgi?p=kernel;a=commitdiff;h=ffddb9a5b8a9dabfcf8fcf0e86884e52c762fef>

Merged 2.6.16.15 from /linux/kernel/git/stable/linux-2.6.16.y

---

--- a/fs/smbfs/dir.c

+++ b/fs/smbfs/dir.c

@@ -434,6 +434,11 @@ smb\_lookup(struct inode \*dir, struct den

if (dentry->d\_name.len > SMB\_MAXNAMELEN)

goto out;

+ /\* Do not allow lookup of names with backslashes in \*/

+ error = -EINVAL;

+ if (memchr(dentry->d\_name.name, '\\', dentry->d\_name.len))

+ goto out;

+

lock\_kernel();

error = smb\_proc\_getattr(dentry, &finfo);

#ifdef SMBFS\_PARANOIA

--- a/include/net/sctp/structs.h

+++ b/include/net/sctp/structs.h

@@ -702,6 +702,7 @@ struct sctp\_chunk {

\_\_u8 tsn\_gap\_acked; /\* Is this chunk acked by a GAP ACK? \*/

\_\_s8 fast\_retransmit; /\* Is this chunk fast retransmitted? \*/

\_\_u8 tsn\_missing\_report; /\* Data chunk missing counter. \*/

+ \_\_u8 data\_accepted; /\* At least 1 chunk in this packet accepted \*/

};

void sctp\_chunk\_hold(struct sctp\_chunk \*);

--- a/net/sctp/inqueue.c

+++ b/net/sctp/inqueue.c

@@ -149,6 +149,7 @@ struct sctp\_chunk \*sctp\_inq\_pop(struct s

/\* This is the first chunk in the packet. \*/

```

chunk->singleton = 1;
ch = (sctp_chunkhdr_t *) chunk->skb->data;
+ chunk->data_accepted = 0;
}

    chunk->chunk_hdr = ch;
--- a/net/sctp/sm_statefuns.c
+++ b/net/sctp/sm_statefuns.c
@@ @ -636,8 +636,9 @@ sctp_disposition_t sctp_sf_do_5_1D_ce(co
 */
    chunk->subh.cookie_hdr =
    (struct sctp_signed_cookie *)chunk->skb->data;
- skb_pull(chunk->skb,
- ntohs(chunk->chunk_hdr->length) - sizeof(sctp_chunkhdr_t));
+ if (!pskb_pull(chunk->skb, ntohs(chunk->chunk_hdr->length) -
+ sizeof(sctp_chunkhdr_t)))
+ goto nomem;

/* 5.1 D) Upon reception of the COOKIE ECHO chunk, Endpoint
 * "Z" will reply with a COOKIE ACK chunk after building a TCB
@@ @ -965,7 +966,8 @@ sctp_disposition_t sctp_sf_beat_8_3(cons
 */
chunk->subh.hb_hdr = (sctp_heartbeathdr_t *) chunk->skb->data;
paylen = ntohs(chunk->chunk_hdr->length) - sizeof(sctp_chunkhdr_t);
- skb_pull(chunk->skb, paylen);
+ if (!pskb_pull(chunk->skb, paylen))
+ goto nomem;

reply = sctp_make_heartbeat_ack(asoc, chunk,
    chunk->subh.hb_hdr, paylen);
@@ @ -1860,8 +1862,9 @@ sctp_disposition_t sctp_sf_do_5_2_4_dupc
 * are in good shape.
*/
    chunk->subh.cookie_hdr = (struct sctp_signed_cookie *)chunk->skb->data;
- skb_pull(chunk->skb, ntohs(chunk->chunk_hdr->length) -
- sizeof(sctp_chunkhdr_t));
+ if (!pskb_pull(chunk->skb, ntohs(chunk->chunk_hdr->length) -
+ sizeof(sctp_chunkhdr_t)))
+ goto nomem;

/* In RFC 2960 5.2.4 3, if both Verification Tags in the State Cookie
 * of a duplicate COOKIE ECHO match the Verification Tags of the
@@ @ -5151,7 +5154,9 @@ static int sctp_eat_data(const struct sc
int tmp;
__u32 tsn;
int account_value;
+ struct sctp_tsnmap *map = (struct sctp_tsnmap *)&asoc->peer.tsn_map;
struct sock *sk = asoc->base.sk;

```

```

+ int rcvbuf_over = 0;

    data_hdr = chunk->subh.data_hdr = (sctp_datahdr_t *)chunk->skb->data;
    skb_pull(chunk->skb, sizeof(sctp_datahdr_t));
@@ -5162,10 +5167,16 @@ static int sctp_eat_data(const struct sc
/* ASSERT: Now skb->data is really the user data. */

/*
- * if we are established, and we have used up our receive
- * buffer memory, drop the frame
+ * If we are established, and we have used up our receive buffer
+ * memory, think about dropping the frame.
+ * Note that we have an opportunity to improve performance here.
+ * If we accept one chunk from an skbuff, we have to keep all the
+ * memory of that skbuff around until the chunk is read into user
+ * space. Therefore, once we accept 1 chunk we may as well accept all
+ * remaining chunks in the skbuff. The data_accepted flag helps us do
+ * that.
*/
- if (asoc->state == SCTP_STATE_ESTABLISHED) {
+ if ((asoc->state == SCTP_STATE_ESTABLISHED) && (!chunk->data_accepted)) {
/*
 * If the receive buffer policy is 1, then each
 * association can allocate up to sk_rcvbuf bytes
@@ -5176,9 +5187,25 @@ static int sctp_eat_data(const struct sc
    account_value = atomic_read(&asoc->rmem_alloc);
    else
        account_value = atomic_read(&sk->sk_rmem_alloc);
-
- if (account_value > sk->sk_rcvbuf)
-     return SCTP_IERROR_IGNORE_TSN;
+ if (account_value > sk->sk_rcvbuf) {
+ /*
+ * We need to make forward progress, even when we are
+ * under memory pressure, so we always allow the
+ * next tsn after the ctsn ack point to be accepted.
+ * This lets us avoid deadlocks in which we have to
+ * drop frames that would otherwise let us drain the
+ * receive queue.
+ */
+ if ((sctp_tsnmap_get_ctsn(map) + 1) != tsn)
+     return SCTP_IERROR_IGNORE_TSN;
+
+ /*
+ * We're going to accept the frame but we should renege
+ * to make space for it. This will send us down that
+ * path later in this function.
+ */

```

```

+ rcvbuf_over = 1;
+ }
}

/* Process ECN based congestion.
@@ -5226,6 +5253,7 @@ static int sctp_eat_data(const struct sc
dataalen -= sizeof(sctp_data_chunk_t);

deliver = SCTP_CMD_CHUNK_ULP;
+ chunk->data_accepted = 1;

/* Think about partial delivery.*/
if ((dataalen >= asoc->rwnd) && (!asoc->ulpq.pd_mode)) {
@@ -5242,7 +5270,8 @@ static int sctp_eat_data(const struct sc
 * large spill over.
*/
if (!asoc->rwnd || asoc->rwnd_over ||
- (dataalen > asoc->rwnd + asoc->frag_point)) {
+ (dataalen > asoc->rwnd + asoc->frag_point) ||
+ rcvbuf_over) {

/* If this is the next TSN, consider renegotiating to make
 * room. Note: Playing nice with a confused sender. A
@@ -5250,8 +5279,8 @@ static int sctp_eat_data(const struct sc
 * space and in the future we may want to detect and
 * do more drastic renegotiation.
*/
- if (sctp_tsnmap_has_gap(&asoc->peer.tsn_map) &&
- (sctp_tsnmap_get_ctsn(&asoc->peer.tsn_map) + 1) == tsn) {
+ if (sctp_tsnmap_has_gap(map) &&
+ (sctp_tsnmap_get_ctsn(map) + 1) == tsn) {
    SCTP_DEBUG_PRINK("Renegotiating for tsn:%u\n", tsn);
    deliver = SCTP_CMD_RENEGE;
} else {
--- a/net/sctp/sm_statetable.c
+++ b/net/sctp/sm_statetable.c
@@ -366,9 +366,9 @@ const sctp_sm_table_entry_t *sctp_sm_loo
/* SCTP_STATE_EMPTY */ \
{.fn = sctp_sf_ootb, .name = "sctp_sf_ootb"}, \
/* SCTP_STATE_CLOSED */ \
- {.fn = sctp_sf_bug, .name = "sctp_sf_bug"}, \
+ {.fn = sctp_sf_discard_chunk, .name = "sctp_sf_discard_chunk"}, \
/* SCTP_STATE_COOKIE_WAIT */ \
- {.fn = sctp_sf_bug, .name = "sctp_sf_bug"}, \
+ {.fn = sctp_sf_discard_chunk, .name = "sctp_sf_discard_chunk"}, \
/* SCTP_STATE_COOKIE_ECHOED */ \
{.fn = sctp_sf_do_ecne, .name = "sctp_sf_do_ecne"}, \
/* SCTP_STATE_ESTABLISHED */ \

```

```

@@ -380,7 +380,7 @@ const sctp_sm_table_entry_t *sctp_sm_loo
 /* SCTP_STATE_SHUTDOWN_RECEIVED */ \
 {.fn = sctp_sf_do_ecne, .name = "sctp_sf_do_ecne"}, \
 /* SCTP_STATE_SHUTDOWN_ACK_SENT */ \
- {.fn = sctp_sf_bug, .name = "sctp_sf_bug"}, \
+ {.fn = sctp_sf_discard_chunk, .name = "sctp_sf_discard_chunk"}, \
 } /* TYPE_SCTP_ECN_ECNE */

#define TYPE_SCTP_ECN_CWR { \
@@ -401,7 +401,7 @@ const sctp_sm_table_entry_t *sctp_sm_loo
 /* SCTP_STATE_SHUTDOWN_RECEIVED */ \
 {.fn = sctp_sf_discard_chunk, .name = "sctp_sf_discard_chunk"}, \
 /* SCTP_STATE_SHUTDOWN_ACK_SENT */ \
- {.fn = sctp_sf_bug, .name = "sctp_sf_bug"}, \
+ {.fn = sctp_sf_discard_chunk, .name = "sctp_sf_discard_chunk"}, \
 } /* TYPE_SCTP_ECN_CWR */

#define TYPE_SCTP_SHUTDOWN_COMPLETE { \
@@ -647,7 +647,7 @@ chunk_event_table_unknown[SCTP_STATE_NUM
 /* SCTP_STATE_EMPTY */ \
 {.fn = sctp_sf_bug, .name = "sctp_sf_bug"}, \
 /* SCTP_STATE_CLOSED */ \
- {.fn = sctp_sf_bug, .name = "sctp_sf_bug"}, \
+ {.fn = sctp_sf_error_closed, .name = "sctp_sf_error_closed"}, \
 /* SCTP_STATE_COOKIE_WAIT */ \
 {.fn = sctp_sf_do_prm_requestheartbeat, \
 .name = "sctp_sf_do_prm_requestheartbeat"}, \
--- a/net/sctp/ulpqueue.c
+++ b/net/sctp/ulpqueue.c
@@ -279,6 +279,7 @@ static inline void sctp_ulpq_store_reasm
 static struct sctp_ulpevent *sctp_make_reassembled_event(struct sk_buff_head *queue, struct
 sk_buff *f_frag, struct sk_buff *l_frag)
{
    struct sk_buff *pos;
+ struct sk_buff *new = NULL;
    struct sctp_ulpevent *event;
    struct sk_buff *pnext, *last;
    struct sk_buff *list = skb_shinfo(f_frag)->frag_list;
@@ -297,11 +298,33 @@ static struct sctp_ulpevent *sctp_make_r
 */
    if (last)
        last->next = pos;
- else
-    skb_shinfo(f_frag)->frag_list = pos;
+ else {
+     if (skb_cloned(f_frag)) {
+         /* This is a cloned skb, we can't just modify
+          * the frag_list. We need a new skb to do that.

```

```

+ * Instead of calling skb_unshare(), we'll do it
+ * ourselves since we need to delay the free.
+ */
+ new = skb_copy(f_frag, GFP_ATOMIC);
+ if (!new)
+     return NULL; /* try again later */
+
+ new->sk = f_frag->sk;
+
+ skb_shinfo(new)->frag_list = pos;
+ } else
+     skb_shinfo(f_frag)->frag_list = pos;
+ }

/* Remove the first fragment from the reassembly queue. */
__skb_unlink(f_frag, queue);
+
+ /* if we did unshare, then free the old skb and re-assign */
+ if (new) {
+     kfree_skb(f_frag);
+     f_frag = new;
+ }
+
while (pos) {

pnext = pos->next;

```

#### File Attachments

---

1) [diff-merge-2.6.16.15-20060510](#), downloaded 380 times

---