strongSwan - Bug #295

charon IKEv1 race receiving response before initiator side checkin of SA [PATCH]

22.02.2013 23:15 - Stephen Rust

Status:	Closed	**Start date:**	22.02.2013
Priority:	Normal	**Due date:**	
Assignee:	Martin Willi	**Estimated time:**	0.00 hour
Category:	libcharon		
Target version:	5.0.3		
Affected version:	5.0.2	**Resolution:**	Fixed

Description

Hi,

I have a test setup with strongswan (5.0.1) on both initiator and remote sides.

I hit a condition during IKEv1 SA negotiation where the response to a message came back so quickly from the remote side, that the initiator side did not have a chance to "checkin" the SA that it had checked out to send its message. This caused the response message to be dropped by the initiator side, and the initiator to then re-send its negotiation message. However, in my case, the response that was dropped happened to be the last message in the IKE SA negotiation, so the remote side considered the IKE SA to be "up". And so it subsequently *dropped* any retries that it received from the initiator because it considered the SA to be up.

So, the remote side thought it was "up", the initiator side continued to retry its final negotiation message (in this case, the [ID HASH] portion of IKE negotiation), and in actuality, the SA was *not* up because the initiator was stuck attempting retry. This prevented the SA from coming up for some timeout period (~2 hours).

You can see this problem illustrated in the log messages as seen from the "initiator" side:
[IP .93 is the initiator]
[IP .201 is the responder]

Feb 19 17:33:41 localhost charon: 01[MGR] checkout IKE_SA by message
Feb 19 17:33:41 localhost charon: 01[MGR] IKE_SA test-conn[4] successfully checked out
Feb 19 17:33:41 localhost charon: 01[NET] received packet: from XXX.XX.XX.201[500] to XXX.XX.XX.93[500]
Feb 19 17:33:41 localhost charon: 01[NET] sending packet: from XXX.XX.XX.93[500] to XXX.XX.XX.201[500]
Feb 19 17:33:41 localhost charon: 08[NET] sending packet: from XXX.XX.XX.93[500] to XXX.XX.XX.201[500]
Feb 19 17:33:41 localhost charon: 01[MGR] checkin IKE_SA test-conn[4]
Feb 19 17:33:41 localhost charon: 07[NET] received packet: from XXX.XX.XX.201[500] to XXX.XX.XX.93[500]
Feb 19 17:33:41 localhost charon: 02[MGR] checkout IKE_SA by message
Feb 19 17:33:41 localhost charon: 02[MGR] ignoring request with ID 0, already processing
Feb 19 17:33:41 localhost charon: 01[MGR] check-in of IKE_SA successful.

Notice the 2nd to last message "ignoring request with ID 0, already processing" comes BEFORE the last message "check-in of IKE_SA successful." So before the initiator could checkin the IKE_SA successfully, it received the response, and code in the receive path ignored the message. A race between threads.

In iterations of my testing when the race does not occur, the log messages show the checkin of IKE_SA clearly before the response message was received:

Feb 19 17:32:53 localhost charon: 13[NET] sending packet: from XXX.XX.XX.93[500] to XXX.XX.XX.201[500]
Feb 19 17:32:53 localhost charon: 08[NET] sending packet: from XXX.XX.XX.93[500] to XXX.XX.XX.201[500]
Feb 19 17:32:53 localhost charon: 13[MGR] checkin IKE_SA test-conn[3]
Feb 19 17:32:53 localhost charon: 13[MGR] check-in of IKE_SA successful.

followed by:

Feb 19 17:32:53 localhost charon: 07[NET] received packet: from XXX.XX.XX.201[500] to XXX.XX.XX.93[500]
Feb 19 17:32:53 localhost charon: 01[MGR] checkout IKE_SA by message
Feb 19 17:32:53 localhost charon: 01[MGR] IKE_SA test-conn[3] successfully checked out

I tracked down in the code to what I think is the problem. I hit this on 5.0.1, but the latest git does not have any code changes in this area. The key wording to me was that it said "ignoring request", when it was the response from the remote side that had been received.

The code where this check comes from is in src/libcharon/sa/ike_sa_manager.c:

```
if (get_entry_by_id(this, id, &entry, &segment)  SUCCESS)
{
                /* only check out in IKEv2 if we are not already processing it */
                if (message->get_request(message) &&
                    message->get_message_id(message)  entry->message_id)
        {
                        DBG1(DBG_MGR, "ignoring request with ID %u, already processing",
                             entry->message_id);
                }
```

In my case, I am using IKEv1. In IKEv1, the value of "is_request" is hardcoded to TRUE when receiving the message. Additionally, the message_id appears to always be 0 for IKEv1. Since there does not seem to a concept of request/response messages in IKEv1, this code seems to indicate that if the IKE_SA is ever checked out processing a previous message, another message received during this time will be ignored. Again, "is_request" will always be TRUE for IKEv1, in the code.

In the IKEv2 case, "is_request" is set according to what type of message was received. Either it was a "request" message, or it was a "response" message. So in the IKEv2 case, this code seems to mean that for requests, we know that strongswan is already processing that request, and so in order to not duplicate work and perform the same operation again, drop the request. We're working on it and will respond when we are done. Perhaps its a duplicate "request" that we are suppressing, in the IKEv2 case.

However, for IKEv1, this check appears to be incorrect. In fact, the very next line of code will *wait* for the checkin of the SA to occur by calling:

```
else if (wait_for_entry(this, entry, segment))
```

If we get past the check which ignores the message, this 'wait_for_entry()' is definitely what we want, in the case that I hit.

Below is the patch to perform the check to 'ignore the request' only for IKEv2. The comment even mentions that it perhaps should only be performed for IKEv2. I'm not sure if this is the best way to fix this, depending on any other parts of the code that I have missed, but this definitely fixes the problem that I have described above for the specific race condition in my example. Please take a look, and if there is a better way to fix this, I would appreciate any feedback or advice on a better way to do so.

Thank you!
Steve

--

```
diff --git a/src/libcharon/sa/ike_sa_manager.c b/src/libcharon/sa/ike_sa_manager.c
--- a/src/libcharon/sa/ike_sa_manager.c
+++ b/src/libcharon/sa/ike_sa_manager.c
@ -1251,7 +1251,8 @ METHOD(ike_sa_manager_t, checkout_by_message, ike_sa_t*,
if (get_entry_by_id(this, id, &entry, &segment)  SUCCESS)        {
/* only check out in IKEv2 if we are not already processing it */
-               if (message->get_request(message) &&
+               if (message->get_major_version(message)  IKEV2_MAJOR_VERSION &&
+                   message->get_request(message) &&
message->get_message_id(message) == entry->message_id)               {
DBG1(DBG_MGR, "ignoring request with ID %u, already processing",
```

Associated revisions

Revision cdf75a39 - 25.02.2013 12:12 - Martin Willi

Move initial message dropping to task manager

When the last request message of the initial tunnel setup is retransmitted,
we must retransmit the response instead of ignoring the request.

Fixes #295.

History

#1 - 25.02.2013 12:22 - Martin Willi

- Category changed from charon to libcharon

- Status changed from New to Feedback

- Target version set to 5.0.3

- Resolution set to Fixed

Hi Steve,

> Below is the patch to perform the check to 'ignore the request' only for IKEv2.

Thanks for the detailed report and your patch.

I think the main issue here is that we don't accept a retransmit of a Main Mode exchange if the ISAKMP SA is established. Fixing that should fix this issue, as the retransmit will bring up the tunnel.

This is not an ideal solution, though. We should avoid the retransmit and process the first message instead. This can be done by fixing that "already processing" check. Using the message ID in Main Mode does not really work, as it is always zero. We should check something else, such as a hash over the full message. Avoiding the check completely is not unproblematic, as it may eat up all your threads if (for example) authentication against RADIUS takes too long and the peer aggressively retransmits.

I've pushed two patches that fixes the issue here. Let me know if this works four you as well.

Best regards
Martin

#2 - 11.03.2013 18:28 - Stephen Rust

Martin,

Thank you for the quick turnaround on this bug report, and how quickly you provided patches. I've been testing the patches since you pushed them in our environment and have had complete success -- they address the problems we were seeing and for which I filed the bug report.

Thank you for providing the correct fix, and the reasoning behind it. Much appreciated.

Steve

#3 - 12.03.2013 09:38 - Martin Willi

- Status changed from Feedback to Closed

#4 - 06.05.2013 18:59 - Andreas Steffen

- Assignee set to Martin Willi