strongSwan - Bug #816

race condition in charon with IKEv2 resulting in lost IKE_SA after IKE_SA_INIT

10.01.2015 17:08 - Imre Keri

Status:	Closed	**Start date:**	10.01.2015
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	charon		
Target version:	5.3.0		
Affected version:	5.1.3	**Resolution:**	Fixed

Description

Hi,

I observed a race condition in charon between threads while it processed an IKE_SA_INIT request. The race is probably triggered by system overload. The IKE_SA_INIT was retransmitted by the remote side one time. The end result is that charon processed both the original and retransmitted IKE_SA_INIT, so the tunnel (IKE_SA) was deleted locally while the remote side thinked it was up. Please see log excerpt below for details.

```
---> first IKE_SA_INIT received
Nov 15 12:06:10 localhost charon: 01[NET] received packet => 432 bytes @ 0x7fc6662c33b0

---> IKE_SA_INIT handled by thread 22
Nov 15 12:06:12 localhost charon: 22[NET] received packet: from 2dec::1001[500] to 2dec::1002[500]
 (432 bytes)
Nov 15 12:06:12 localhost charon: 22[ENC] parsed IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(N
ATD_D_IP) ]

---> IKE_SA_INIT response sent out (note high processing time due to system overload):
Nov 15 12:06:17 localhost charon: 03[NET] sending packet: from 2dec::1002[500] to 2dec::1001[500]

---> retransmitted IKE_SA_INIT coming in (this was sent by remote side before it got the response,
 and then it was sitting in local socket input queue):
Nov 15 12:06:17 localhost charon: 01[NET] received packet => 432 bytes @ 0x7fc6662c33b0

---> In the meantime the remote side received the IKE_SA_INIT response and sent an IKE_AUTH reques
t.
---> IKE_AUTH coming in:
Nov 15 12:06:19 localhost charon: 01[NET] received packet => 444 bytes @ 0x7fc6662c33b0

---> IKE_AUTH is processed:
Nov 15 12:06:20 localhost charon: 21[NET] received packet: from 2dec::1001[500] to 2dec::1002[500]
 (444 bytes)
Nov 15 12:06:20 localhost charon: 21[ENC] parsed IKE_AUTH request 1 [ IDi IDr AUTH SA TSi TSr N(MU
LT_AUTH) N((40960)) ]

---> IKE_AUTH response sent back:
Nov 15 12:06:30 localhost charon: 21[ENC] generating IKE_AUTH response 1 [ IDr AUTH SA TSi TSr ]
Nov 15 12:06:30 localhost charon: 03[NET] sending packet: from 2dec::1002[500] to 2dec::1001[500]

---> retransmitted IKE_SA_INIT is processed -- this should not happen! the order has now changed.
Nov 15 12:06:30 localhost charon: 11[NET] received packet: from 2dec::1001[500] to 2dec::1002[500]
 (432 bytes)

---> As a result the local tunnel is cleared, and the other side will just see a response and thin
k everything is OK.
Nov 15 12:06:30 localhost charon: 11[IKE] received message ID 0, expected 2. Ignored
Nov 15 12:06:30 localhost charon: 11[IKE] IKE_SA tunnel_autotest_ipv6_0[38] state change: ESTABLIS
HED => DESTROYING
```

I took some looks at the code and interpret the logs like this: The second IKE_SA_INIT was scheduled to thread 11, but before this thread had a chance to check out the SA, the other thread 21 checked out the SA and started to process the IKE_AUTH message. So even though the jobs were scheduled in the right order, they were scheduled to 2 different threads, and nothing prevented the thread processing the IKE_AUTH job to start before the IKE_SA_INIT handler thread. This might happen under heavy overload only, but seems like a real problem nevertheless.

Another related issue can be the exception with IKE_SA_INIT messages in task_manager_v2.c after the "received message ID %d, expected %d. Ignored" printout. Without this check, what is not present in strongswan 4, this case would probably work fine.

Can you provide an opinion or fix/workaround for this issue?

Regards,
Imre Keri

Associated revisions

Revision eb251906 - 04.03.2015 13:47 - Tobias Brunner

ikev2: Don't destroy the SA if an IKE_SA_INIT with unexpected MID is received

This reverts 8f727d800751 ("Clean up IKE_SA state if IKE_SA_INIT request
does not have message ID 0") because it allowed to close any IKE_SA by
sending an IKE_SA_INIT with an unexpected MID and both SPIs set to those
of that SA.

The next commit will prevent SAs from getting created for IKE_SA_INIT messages
with invalid MID.

Fixes #816.

Revision 650a3ad5 - 04.03.2015 13:47 - Tobias Brunner

ike-sa-manager: Make sure the message ID of initial messages is 0

It is mandated by the RFCs and it is expected by the task managers.

Initial messages with invalid MID will be treated like regular messages,
so no IKE_SA will be created for them. Instead, if the responder SPI is 0
no SA will be found and the message is rejected with ALERT_INVALID_IKE_SPI.
If an SPI is set and we do find an SA, then we either ignore the message
because the MID is unexpected, or because we don't allow initial messages
on established connections.

There is one exception, though, if an attacker can slip in an IKE_SA_INIT
with both SPIs set before the client's IKE_AUTH is handled by the server,
it does get processed (see next commit).

References #816.

Revision dd0ebb54 - 04.03.2015 13:47 - Tobias Brunner

ikev2: Only accept initial messages in specific states

The previous code allowed an attacker to slip in an IKE_SA_INIT with
both SPIs and MID 1 set when an IKE_AUTH would be expected instead.

References #816.

History

#1 - 11.02.2015 14:14 - Tobias Brunner

- Description updated

- Status changed from New to Feedback

- Assignee set to Tobias Brunner

```
---> retransmitted IKE_SA_INIT is processed -- this should not happen! the order has now changed.
Nov 15 12:06:30 localhost charon: 11[NET] received packet: from 2dec::1001[500] to 2dec::1002[500] (432 by
tes)
```

This is not a problem. The IKE_SA_INIT retransmit could also be delayed on the network and arrive after the IKE_AUTH request. So charon should be able handle such a situation anyway.

```
---> As a result the local tunnel is cleared, and the other side will just see a response and think everyt
hing is OK.
Nov 15 12:06:30 localhost charon: 11[IKE] received message ID 0, expected 2. Ignored
Nov 15 12:06:30 localhost charon: 11[IKE] IKE_SA tunnel_autotest_ipv6_0[38] state change: ESTABLISHED => D
ESTROYING
```

As you point out it looks like the check here source:src/libcharon/sa/ikev2/task_manager_v2.c#L1340 is in fact problematic if the IKE_SA_INIT message is delayed. The check above it, that looks for retransmits, only checks for expected MID - 1, so since the IKE_AUTH message has been handled already we now expect the next MID to be 2 (hence a retransmit of the IKE_AUTH request with MID 1 would be fine, but a retransmit of the IKE_SA_INIT message with MID 0 ends in that *else* block).

I suppose we could do something like this:

```
diff --git a/src/libcharon/sa/ikev2/task_manager_v2.c b/src/libcharon/sa/ikev2/t
index e9a677a655e4..69118abe78df 100644
--- a/src/libcharon/sa/ikev2/task_manager_v2.c
+++ b/src/libcharon/sa/ikev2/task_manager_v2.c
@@ -1339,7 +1339,7 @@ METHOD(task_manager_t, process_message, status_t,
        {
            DBG1(DBG_IKE, "received message ID %d, expected %d. Ignored",
                 mid, this->responding.mid);
-           if (msg->get_exchange_type(msg) == IKE_SA_INIT)
+           if (mid != 0 && msg->get_exchange_type(msg) == IKE_SA_INIT)
            {   /* clean up IKE_SA state if IKE_SA_INIT has invalid msg ID */
                return DESTROY_ME;
            }
```

This makes the *if* statement actually do what the comment there indicates, i.e. make sure the MID of the IKE_SA_INIT is valid, while still allowing us to ignore retransmits for it after IKE_AUTH has been handled.

#2 - 18.02.2015 11:25 - Imre Keri

Thanks for looking into this!

Tobias Brunner wrote:

> > [...]
>
> This is not a problem. The IKE_SA_INIT retransmit could also be delayed on the network and arrive after the IKE_AUTH request. So charon should be able handle such a situation anyway.

My concern was that charon could add additional shuffle to the messages besides the network. But I agree it could happen anyway, so this is not the real problem.

> > [...]
>
> As you point out it looks like the check here source:src/libcharon/sa/ikev2/task_manager_v2.c#L1340 is in fact problematic if the IKE_SA_INIT message is delayed. The check above it, that looks for retransmits, only checks for expected MID - 1, so since the IKE_AUTH message has been handled already we now expect the next MID to be 2 (hence a retransmit of the IKE_AUTH request with MID 1 would be fine, but a retransmit of the IKE_SA_INIT message with MID 0 ends in that *else* block).
>
> I suppose we could do something like this:
>
> [...]
>
> This makes the *if* statement actually do what the comment there indicates, i.e. make sure the MID of the IKE_SA_INIT is valid, while still allowing us to ignore retransmits for it after IKE_AUTH has been handled.

As far as my (basic) knowledge goes, this seems like a good solution. The RFC states that the IKE_SA_INIT "will" be always 0, so any other mid is invalid, and the reordering issue can only happen with IKE_SA_INIT with message ID 0. (After the IKE_SA is destroyed, is there any chance that a new IKE_SA is created as result of the invalid message? Then it would end up in a situation when a nonzero IKE_SA_INIT could be the message to be dropped...?)

Will this be included in a future charon version?

Regards,
Imre Keri

#3 - 18.02.2015 15:46 - Tobias Brunner

- Tracker changed from Issue to Bug

- Target version set to 5.3.0

> (After the IKE_SA is destroyed, is there any chance that a new IKE_SA is created as result of the invalid message? Then it would end up in a situation when a nonzero IKE_SA_INIT could be the message to be dropped...?)

What do you mean?

> Will this be included in a future charon version?

Yes, this will be included in the next release.

#4 - 24.02.2015 13:00 - Imre Keri

Tobias Brunner wrote:

>> (After the IKE_SA is destroyed, is there any chance that a new IKE_SA is created as result of the invalid message? Then it would end up in a situation when a nonzero IKE_SA_INIT could be the message to be dropped...?)
>
> What do you mean?

I was thinking that the invalid IKE_SA_INIT results in the IKE_SA to get deleted. Then if the sender is retransmitting the same IKE_SA_INIT, a new IKE_SA might be created as result of this message, what is really invalid then since the IKE_SA_INIT had non 0 message ID. So an extra safety check could be to not create an IKE_SA if the message has nonzero message ID, for example in the checkout_by_message function (source:src/libcharon/sa/ike_sa_manager.c#L1170). But I might have misunderstood something.

#5 - 24.02.2015 13:25 - Tobias Brunner

>>> (After the IKE_SA is destroyed, is there any chance that a new IKE_SA is created as result of the invalid message? Then it would end up in a situation when a nonzero IKE_SA_INIT could be the message to be dropped...?)
>>
>> What do you mean?
>
> I was thinking that the invalid IKE_SA_INIT results in the IKE_SA to get deleted. Then if the sender is retransmitting the same IKE_SA_INIT, a new IKE_SA might be created as result of this message, what is really invalid then since the IKE_SA_INIT had non 0 message ID.

If the IKE_SA_INIT is in fact invalid (i.e. the MID is != 0) we will end up in the exact same code path. After creating the IKE_SA object process_message() will be called on the task manager, which will reject the message and the SA gets immediately destroyed again. An IKE_SA_INIT that is valid, but is delayed, could cause another IKE_SA to get created, though, but if the initiator does not react to the IKE_SA_INIT response it will get destroyed after the half-open timeout.

> So an extra safety check could be to not create an IKE_SA if the message has nonzero message ID, for example in the checkout_by_message function (source:src/libcharon/sa/ike_sa_manager.c#L1170).

I guess such a check could be added to improve the performance in such a situation. But while an invalid IKE_SA_INIT gets parsed and some state for the IKE_SA is temporarily allocated, we won't process the message, for instance, we don't do DH. so attacking a server with valid IKE_SA_INITs is probably more efficient as it will cause it to do DH and keep the allocated state around until it times out (DoS protection is enabled by default, though).

#6 - 24.02.2015 16:20 - Imre Keri

OK, I see your point. I can only add that someone might still try such a DoS attack even if it is not efficient. But I'm perfectly fine with your originally provided fix.

#7 - 04.03.2015 13:50 - Tobias Brunner

After thinking about this some more I actually *do* think we have to fix this differently. The problem is that with the current (or even the changed) code it is possible to close any IKE_SA, of which the SPIs are known, by sending an IKE_SA_INIT request with an invalid MID and *both* SPIs set. Such a message would not be treated as "intial" message in checkout_by_message() but would get handed over to the IKE_SA. While we do have a check that doesn't allow IKE_SA_INIT and IKE_AUTH messages once the SA is established, that check is only triggered if the MID matches what we expect. So we would end up in the code path discussed here and destroy the IKE_SA.

I pushed the associated commits to master, which remove this check for IKE_SA_INIT, instead ensure in checkout_by_message() that the MID of initial messages is 0, and don't handle IKE_SA_INIT messages when an IKE_AUTH is expected.

#8 - 24.03.2015 12:38 - Tobias Brunner

- Status changed from Feedback to Closed

#9 - 24.03.2015 12:38 - Tobias Brunner

- Resolution set to Fixed