strongSwan - Bug #1236

IKEV1 conn with the same vpn IP has rekey issue with charon.reuse_ikesa=no

14.12.2015 13:33 - heidi rao

Status:	Closed	**Start date:**	14.12.2015
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	charon		
Target version:	5.4.0		
Affected version:	5.3.5	**Resolution:**	Fixed

Description

Hi,
strongswan.conf is as below:

```
include strongswan.d/*.conf

charon {
    load_modular = yes
    plugins {
        include strongswan.d/charon/*.conf
    }
    reuse_ikesa=no
}
```

I config two connections as below:

```
conn rule1~v1
        rekeymargin=500
        rekeyfuzz=100%
        keyexchange=ikev1
        left=35.6.6.205
        right=35.6.6.105
        leftsubnet=35.6.6.207/32
        rightsubnet=35.6.6.107/32
        authby=secret
        leftid=35.6.6.205
        rightid=%any
        ike=aes128-sha1-modp768!
        esp=aes128-sha1!
        type=tunnel
        ikelifetime=5000s
        keylife=5000s
        mobike=no
        auto=route
        reauth=no

conn rule2~v1
        rekeymargin=500
        rekeyfuzz=100%
        keyexchange=ikev1
        left=35.6.6.205
        right=35.6.6.105
        leftsubnet=35.6.6.209/32
        rightsubnet=35.6.6.109/32
        authby=secret
        leftid=35.6.6.205
        rightid=%any
        ike=aes128-sha1-modp768!
        esp=aes128-sha1!
        type=tunnel
        ikelifetime=5000s
        keylife=5000s
        mobike=no
        auto=route
        reauth=no
```

These two connection established:

```
Dec 14 18:55:13 24F-VFPC-006 charon: 11[IKE] IKE_SA rule1~v1[1] established between 35.6.6.105[35.
6.6.105]...35.6.6.205[35.6.6.205]
Dec 14 18:55:13 24F-VFPC-006 charon: 13[IKE] CHILD_SA rule1~v1{3} established with SPIs c56e6c6f_i
 c17ac527_o and TS 35.6.6.107/32 === 35.6.6.207/32
Dec 14 18:55:19 24F-VFPC-006 charon: 10[IKE] IKE_SA rule1~v1[2] established between 35.6.6.105[35.
6.6.105]...35.6.6.205[35.6.6.205]
Dec 14 18:55:19 24F-VFPC-006 charon: 02[IKE] CHILD_SA rule2~v1{4} established with SPIs ceb39d2f_i
 c0020abc_o and TS 35.6.6.109/32 === 35.6.6.209/32
```

Then the peer trigger the child_sa rekey, but it can't find the CHILD_SA, this will leading to

```
Dec 14 18:56:13 24F-VFPC-006 charon: 14[IKE] received DELETE for ESP CHILD_SA with SPI c17ac527
Dec 14 18:56:13 24F-VFPC-006 charon: 14[IKE] CHILD_SA not found, ignored
```

I test it on 5.3.2 and 5.3.5, both of them has this issue.

Associated revisions

Revision 24ab8530 - 01.02.2016 11:37 - Tobias Brunner

ikev1: Always enable charon.reuse_ikesa

With IKEv1 we have to reuse IKE_SAs as otherwise the responder might
detect the new SA as reauthentication and will "adopt" the CHILD_SAs of
the original IKE_SA, while the initiator will not do so. This could
cause CHILD_SA rekeying to fail later.

Fixes #1236.

History

#1 - 14.12.2015 13:49 - heidi rao

If the local one initial the rekey, it will print:

```
Dec 14 20:45:25 24F-VFPC-006 charon: 05[IKE] closing expired CHILD_SA rule1~v2{3} with SPIs c50f94a3_i c3593d6
5_o and TS 35.6.6.107/32 === 35.6.6.207/32
Dec 14 20:45:25 24F-VFPC-006 charon: 05[IKE] sending DELETE for ESP CHILD_SA with SPI c50f94a3
Dec 14 20:45:25 24F-VFPC-006 charon: 07[JOB] CHILD_SA ESP/0xc3593d65/35.6.6.205 not found for delete
Dec 14 20:45:30 24F-VFPC-006 charon: 02[IKE] closing expired CHILD_SA rule2~v2{4} with SPIs c7159b3d_i cf5ff25
3_o and TS 35.6.6.109/32 === 35.6.6.209/32
Dec 14 20:45:30 24F-VFPC-006 charon: 02[IKE] sending DELETE for ESP CHILD_SA with SPI c7159b3d
Dec 14 20:45:30 24F-VFPC-006 charon: 11[JOB] CHILD_SA ESP/0xcf5ff253/35.6.6.205 not found for delete
```

However, the peer only receive one DELETE, the other one seems not received.

```
Dec 14 20:45:30 CLA-0 charon[17812]: 10[IKE] received DELETE for ESP CHILD_SA with SPI c7159b3d
Dec 14 20:45:30 CLA-0 charon[17812]: 10[IKE] closing CHILD_SA rule2~v2{2} with SPIs cf5ff253_i (304 bytes) c71
59b3d_o (168 bytes) and TS 35.6.6.209/32 === 35.6.6.109/32
Dec 14 20:45:30 CLA-0 charon[17812]: 10[IKE] closing CHILD_SA rule2~v2{2} with SPIs cf5ff253_i (304 bytes) c71
59b3d_o (168 bytes) and TS 35.6.6.209/32 === 35.6.6.109/32
```

#2 - 14.12.2015 15:35 - Tobias Brunner

- Description updated

- Status changed from New to Feedback

- Priority changed from High to Normal

What device is running on the other end? It might be one of those that always uses the latest IKE SA with the same host for all communication related to any CHILD_SAs between them. You could try enabling *charon.reuse_ikesa* so there won't be another IKE_SA for the second CHILD_SA.

#3 - 14.12.2015 16:01 - heidi rao

Hi,
I find that when peer initialize the rekey, it send INFORMATIONA (DELETE CHILD_SA rule2~v1{4}) using the IKE_SA rule1~v1[1]. However, the peer also configured with reuse_ikesa=no, it seems it doesn't take effect. The peer version is 5.0.1.

#4 - 14.12.2015 16:17 - Tobias Brunner

> I find that when peer initialize the rekey, it send INFORMATIONA (DELETE CHILD_SA rule2~v1{4}) using the IKE_SA rule1~v1[1]. However, the peer also configured with reuse_ikesa=no, it seems it doesn't take effect. The peer version is 5.0.1.

Are you saying you use strongSwan on the other end? And that very old version? (IKEv1 was added with 5.0.0 to charon, so it's definitely possible the behavior has changed since then, in particular with some of the updates regarding reqids in 5.3.0). You should compare the logs of both ends to see what each of them does.

#5 - 14.12.2015 17:25 - heidi rao

- File 5.3.2 added

- File 5.3.5 added

- File ipsec.conf-5.3.2 added

- File ipsec.conf-5.3.5 added

Hi,
I also test the same scenario between 5.3.5 and 5.3.2.
5.3.2 initilize the IKE_SA rule1~v2[1] to 5.3.5, IKE_SA and CHILD_SA created successfully

```
Dec 15 00:05:51 24F-VFPC-004 charon: 05[IKE] initiating Main Mode IKE_SA rule1~v2[1] to 35.6.6.105
Dec 15 00:05:51 24F-VFPC-004 charon: 07[IKE] IKE_SA rule1~v2[1] established between 35.6.6.115[35.6.6.115]...35.6.6.105[35.6.6.105]
Dec 15 00:05:51 24F-VFPC-004 charon: 09[IKE] CHILD_SA rule1~v2{3} established with SPIs c772eb51_i cc9b9624_o and TS 35.6.6.117/32 === 35.6.6.107/32
```

5.3.2 initilize the IKE_SA rule1~v2[2] to 5.3.5, IKE_SA and CHILD_SA created successfully

```
Dec 15 00:06:08 24F-VFPC-004 charon: 06[IKE] initiating Main Mode IKE_SA rule1~v2[2] to 35.6.6.105
Dec 15 00:06:08 24F-VFPC-004 charon: 16[IKE] IKE_SA rule1~v2[2] established between 35.6.6.115[35.6.6.115]...35.6.6.105[35.6.6.105]
Dec 15 00:06:08 24F-VFPC-004 charon: 09[IKE] CHILD_SA rule2~v2{4} established with SPIs c4247086_i c03bf75c_o and TS 35.6.6.119/32 === 35.6.6.109/32
```

However, immediatly after rule1~v2[2] and rule2~v2{4} created, 5.3.5 initialize deleting IKE_SA. It looks abnormal.

```
Dec 15 00:06:18 24F-VFPC-006 charon: 02[IKE] deleting IKE_SA rule1~v2[1] between 35.6.6.105[35.6.6.105]...35.6.6.115[35.6.6.115]
Dec 15 00:06:51 24F-VFPC-006 charon: 12[IKE] closing expired CHILD_SA rule1~v2{3} with SPIs cc9b9624_i c772eb51_o and TS 35.6.6.107/32 === 35.6.6.117/32
```

#6 - 14.12.2015 17:32 - Tobias Brunner

> However, immediatly after rule1~v22 and rule2~v2{4} created, 5.3.5 initialize deleting IKE_SA. It looks abnormal.

No, that's expected. You configured *keylife=60s* (aka *lifetime*) and *rekeymargin=500* (aka *margintime*). Please check the formula at ExpiryRekey.

#7 - 14.12.2015 17:34 - heidi rao

Correct my typo

However, immediatly after rule1~v22 and rule2~v2{4} created, 5.3.5 initialize deleting IKE_SA. It looks abnormal.

```
Dec 15 00:06:18 24F-VFPC-006 charon: 02[IKE] deleting IKE_SA rule1~v2[1] between 35.6.6.105[35.6.6.105]...35.6.6.115[35.6.6.115]
Dec 15 00:06:18 24F-VFPC-006 charon: 02[IKE] IKE_SA rule1~v2[1] state change: ESTABLISHED => DELETING
Dec 15 00:06:18 24F-VFPC-006 charon: 02[IKE] IKE_SA rule1~v2[1] state change: DELETING => DESTROYING
```

5.3.2 delete the IKE_SA and CHILD_SA

```
Dec 15 00:06:18 24F-VFPC-004 charon: 15[IKE] deleting IKE_SA rule1~v2[1] between 35.6.6.115[35.6.6.115]...35.6.6.105[35.6.6.105]
Dec 15 00:06:18 24F-VFPC-004 charon: 15[IKE] IKE_SA rule1~v2[1] state change: ESTABLISHED => DELETING
Dec 15 00:06:18 24F-VFPC-004 charon: 15[IKE] IKE_SA rule1~v2[1] state change: DELETING => DELETING
Dec 15 00:06:18 24F-VFPC-004 charon: 15[IKE] IKE_SA rule1~v2[1] state change: DELETING => DESTROYING
Dec 15 00:06:18 24F-VFPC-004 charon: 15[KNL] deleting SAD entry with SPI c772eb51  (mark 0/0x00000000)
Dec 15 00:06:18 24F-VFPC-004 charon: 15[KNL] deleted SAD entry with SPI c772eb51 (mark 0/0x00000000)
Dec 15 00:06:18 24F-VFPC-004 charon: 15[KNL] deleting SAD entry with SPI cc9b9624  (mark 0/0x00000000)
Dec 15 00:06:18 24F-VFPC-004 charon: 15[KNL] deleted SAD entry with SPI cc9b9624 (mark 0/0x00000000)
```

Then, rekey start on 5.3.5

```
Dec 15 00:06:51 24F-VFPC-006 charon: 12[IKE] closing expired CHILD_SA rule1~v2{3} with SPIs cc9b9624_i c772eb51_o and TS 35.6.6.107/32
```

```
=== 35.6.6.117/32
Dec 15 00:07:08 24F-VFPC-006 charon: 05[IKE] closing expired CHILD_SA rule2~v2{4} with SPIs c03bf75c_i c4247086_o and TS 35.6.6.109/32 ===
35.6.6.119/32
```

#8 - 14.12.2015 17:39 - Tobias Brunner

> Correct my typo

What typo?

> However, immediatly after rule1~v22 and rule2~v2{4} created, 5.3.5 initialize deleting IKE_SA. It looks abnormal.

Fix your config.

#9 - 14.12.2015 17:41 - heidi rao

keylife just take effect to CHILD_SA life not IKE_SA life, right?
I just change rekeymargin to 10 on both side, however, the result is the same.

#10 - 16.12.2015 13:23 - heidi rao

Hi,
I found that when the second IKE is under negotiation, the responder will adopt the CHILD_SA from other IKE_SA, but I don't understand why it's designed like this. Can you please explain about it?

```
Dec 16 11:31:16 CLA-0 charon[8394]: 15[MGR] checkout IKE_SA
Dec 16 11:31:16 CLA-0 charon[8394]: 15[MGR] IKE_SA rule1~v2[1] successfully checked out
Dec 16 11:31:16 CLA-0 charon[8394]: 15[IKE] detected reauth of existing IKE_SA, adopting 1 children
Dec 16 11:31:16 CLA-0 charon[8394]: 15[IKE] IKE_SA rule1~v2[1] state change: ESTABLISHED => DELETING
Dec 16 11:31:16 CLA-0 charon[8394]: 15[MGR] checkin and destroy IKE_SA rule1~v2[1]
Dec 16 11:31:16 CLA-0 charon[8394]: 15[IKE] IKE_SA rule1~v2[1] state change: DELETING => DESTROYING
Dec 16 11:31:16 CLA-0 charon[8394]: 15[MGR] check-in and destroy of IKE_SA successful
```

Code is in main_mode.c
@case MM_AUTH: {
....

```
if (!this->ph1->build_auth(this->ph1, this->method, message,
                                                                           id_payload->get_encoded(id_payload)
))
{
                                ...
                                default:
                                        if (charon->ike_sa_manager->check_uniqueness(
                                                                        charon->ike_sa_manager, this->ike_sa, FALSE))
                {
                                                DBG1(DBG_IKE, "cancelling Main Mode due to uniqueness "
                                                         "policy");
                                                return send_notify(this, AUTHENTICATION_FAILED);
                                        }
                                        if (!establish(this))
                {
                                                return send_notify(this, AUTHENTICATION_FAILED);
                                        }
                                        job = adopt_children_job_create(
                                                                                        this->ike_sa->get_id(t
his->ike_sa));
                                        break;
                        }
                        ....
                        if (job)
{
                                lib->processor->queue_job(lib->processor, (job_t*)job);
                        }
                        return SUCCESS;
                }@
```

#11 - 16.12.2015 14:03 - Tobias Brunner

> I found that when the second IKE is under negotiation, the responder will adopt the CHILD_SA from other IKE_SA, but I don't understand why it's designed like this. Can you please explain about it?

I see. The log message basically says it all:

> Dec 16 11:31:16 CLA-0 charon[8394]: 15[IKE] detected reauth of existing IKE_SA, adopting 1 children

The responder thinks the new IKE_SA is created to reauthenticate (rekey) the first IKE_SA. I guess this means you can't use *charon.reuse_ikesa=no* with IKEv1.

#12 - 17.12.2015 03:10 - heidi rao

It seems the limitation of ikev1, we should always reuse the IKE_SA, then would this check point be added when initialize a IKE_SA? In ike_sa_manage.c:1388
if (!this->reuse_ikesa && (IKEV2 == peer_cfg->get_ike_version(peer_cfg)))

It will be fixed in later release? Thanks!

#13 - 18.12.2015 15:30 - Tobias Brunner

- Tracker changed from Issue to Bug

- Subject changed from IKEV1 conn with the same vpn IP has rekey issue to IKEV1 conn with the same vpn IP has rekey issue with charon.reuse_ikesa=no

- Target version set to 5.4.0

> It seems the limitation of ikev1, we should always reuse the IKE_SA, then would this check point be added when initialize a IKE_SA? In ike_sa_manage.c:1388
> if (!this->reuse_ikesa && (IKEV2 == peer_cfg->get_ike_version(peer_cfg)))
>
> It will be fixed in later release? Thanks!

Yes, something like that makes sense. I pushed a commit to the *1236-reuse-ikesa-ikev1* branch.

#14 - 01.02.2016 11:38 - Tobias Brunner

- Category set to charon

- Status changed from Feedback to Closed

- Assignee set to Tobias Brunner

- Resolution set to Fixed

Files

ipsec.conf	1.09 KB	14.12.2015	heidi rao
rekey_issue	16.7 KB	14.12.2015	heidi rao
strongswan.conf	299 Bytes	14.12.2015	heidi rao
5.3.5	49.1 KB	14.12.2015	heidi rao
5.3.2	51.6 KB	14.12.2015	heidi rao
ipsec.conf-5.3.2	1.1 KB	14.12.2015	heidi rao
ipsec.conf-5.3.5	1.09 KB	14.12.2015	heidi rao