strongSwan - Bug #2828

HA plugin: pool segmentation issue ?

20.11.2018 07:52 - Frédéric Martinsons

Status:	Closed	**Start date:**	20.11.2018
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	high availability (ha plugin)		
Target version:	5.7.2		
Affected version:	5.7.1	**Resolution:**	Fixed

Description

Hello,

we have currently a strongswan HA setup with 12 cluserIP segments (charon.plugins.ha.segment_count = 12) and it works great.
But looking at the ha_attribute code (we are on a 5.5.3 but this part hadn't changed since the beginning) inside the acquire_address method, we can see we get segment only for 8 differents bits (0 to 7), this will lead to only use 6 out of 12 segments.
We can see this by a simple loop in (ha_attribute.c) like:

```
for(int bit = 0; bit < 8; bit++)
{
    printf("%u -> %u\n",bit,this->kernel->get_segment_int(this->kernel,bit);
}
```

which gives

0 -> 1
1 -> 5
2 -> 7
3 -> 9
4 -> 2
5 -> 6
6 -> 5
7 -> 2

Moreover I don't understand why the segment test for responsability (by use of responsible_for function) will not be the same as the one that will stands for the IP address we just acquired.

For example the second address that is acquired (from a pool like 10.2.1.0/28) would be 10.2.1.2.
This address gives (via this->kernel->get_segment(this->kernel,host_create_from_string("10.2.1.2",0)) segment 4 while the bit used to calculate offset in pool (2) gives segment 7.

So I have two questions:

- should not 'responsible_for(this,bit)' be replaced by something like 'responsible_for(this,byte*8 + bit)' ?
- why segments calculated with bit is not the same calculated with address (although, address is derivated from bit) ?

Associated revisions

Revision 13f92f64 - 07.12.2018 10:16 - Tobias Brunner

Merge branch 'ha-pool-offset'

Ensure an even distribution of a pool's addresses among all segments.

Fixes #2828.

History

#1 - 20.11.2018 13:25 - Tobias Brunner

- Status changed from New to Feedback

- Target version set to 5.7.2

- should not 'responsible_for(this,bit)' be replaced by something like 'responsible_for(this,byte*8 + bit)' ?

Using more than just the bit definitely makes sense.

- why segments calculated with bit is not the same calculated with address (although, address is derivated from bit) ?

I guess the idea is to just partition the pool. Doing that differently should be fine (as long as all nodes do it the same way) because segments for IKE/ESP are assigned based on the public IP and not the virtual IPs. Also, ha_kernel_t::get_segment() does not support IPv6 addresses, which are allowed for virtual IPs (although we could probably just use the lower 32 bit because the pools are limited to 24 host bits anyway).

However, due to how the segments are mapped, via hashing and hash2segment(), neither approach (using the offset or the actual address) will guarantee that all segments will be mapped uniformly. In your example with a /28 subnet and 12 segments and using the full offset, the 14 addresses (the first and last won't be assigned) will be mapped as follows to the segments (i.e. it's slightly better than using just the bit):

```
bit only      offset
 1 -> 1        1 -> 1
 2 -> 3        2 -> 2
 3 -> 0        3 -> 0
 4 -> 0        4 -> 1
 5 -> 4        5 -> 3
 6 -> 2        6 -> 2
 7 -> 2        7 -> 2
 8 -> 0        8 -> 1
 9 -> 2        9 -> 1
10 -> 0       10 -> 1
11 -> 0       11 -> 0
12 -> 0       12 -> 0
```

This improves with larger pools, e.g. with /24:

```
bit only      offset
 1 -> 31       1 -> 19
 2 -> 63       2 -> 20
 3 -> 0        3 -> 19
 4 -> 0        4 -> 19
 5 -> 64       5 -> 25
 6 -> 32       6 -> 23
 7 -> 32       7 -> 23
 8 -> 0        8 -> 33
 9 -> 32       9 -> 19
10 -> 0       10 -> 22
11 -> 0       11 -> 18
12 -> 0       12 -> 14
```

The same with just 4 segments:

```
bit only      offset
 1 -> 94       1 -> 58
 2 -> 96       2 -> 67
 3 -> 64       3 -> 75
 4 -> 0        4 -> 54
```

So not totally uniform, but still quite an improvement.

Anyway, I pushed a change that uses the offset for the segment check to the *2828-ha-pool-offset* branch.

#2 - 20.11.2018 13:50 - Frédéric Martinsons

Ok thanks, I understand there is no optimal solution when using this kind of hashing function.
It may be a naive question but, if the goal is to simply equally partition the pool, why not just dividing the address pool with the number of segment ?
So, in my 12 segments example on a /28 subnet, segment from 1 to 11 will have 1 address (1/12th of 14 addresses) and segment 12 will have 2 adresses.

Do you think it is something that can achieved without degrading HA mechanism ?

#3 - 20.11.2018 16:49 - Tobias Brunner

> It may be a naive question but, if the goal is to simply equally partition the pool, why not just dividing the address pool with the number of

segment ?
So, in my 12 segments example on a /28 subnet, segment from 1 to 11 will have 1 address (1/12th of 14 addresses) and segment 12 will have 2 adresses.

Do you think it is something that can achieved without degrading HA mechanism ?

I guess so (I hope I don't overlook anything). With the full offset as input, we could now simply do offset % <segment count> + 1 to get the segment number. I pushed this to the mentioned branch. For your example this gives you:

```
 1 -> 1
 2 -> 2
 3 -> 2
 4 -> 1
 5 -> 1
 6 -> 1
 7 -> 1
 8 -> 1
 9 -> 1
10 -> 1
11 -> 1
12 -> 1
```

And /24 with 4 segments results in:

```
 1 -> 63
 2 -> 64
 3 -> 64
 4 -> 63
```

#4 - 20.11.2018 17:39 - Frédéric Martinsons

Thank you very much, I'll test these commits on our HA test setup and I'll come back to you as soon as I have results (around 2 weeks I think).
Thanks again.

#5 - 06.12.2018 18:43 - Frédéric Martinsons

Hello, we ran our non reg campaign towards the commits you gave about segments allocation and everything worked as expected.
Thank you very much for your reactivity and your support.

#6 - 07.12.2018 10:41 - Tobias Brunner

- Status changed from Feedback to Closed

- Assignee set to Tobias Brunner

- Resolution set to Fixed

Thanks for the tests, applied to master.