strongSwan - Issue #841

can I implement this to not assign latest used IP address?

03.02.2015 09:58 - richard hu

Status:	Closed		
Priority:	Normal		
Assignee:	Tobias Brunner		
Category:	libcharon		
Affected version:	5.2.2	**Resolution:**	

Description

Current IP address assign is follow the rule to use last time used IP.
Now I want change it to not use the IP address that used recently.
Is this easy to change several lines of code? And which file or function I need to change?
This is only for we internal usage.

Associated revisions

Revision 7d02f8db - 19.03.2015 09:55 - Tobias Brunner

mem-pool: Remove entries without online or offline leases

This avoids filling up the hash table with unused/old identities.

References #841.

History

#1 - 11.02.2015 15:19 - Tobias Brunner

- Status changed from New to Feedback

- Assignee deleted (Martin Willi)

- Priority changed from High to Normal

Are you using the in-memory IP pools, i.e. via *rightsourceip=<subnet>*? If so, you could try to patch source:src/libhydra/attributes/mem_pool.c like this:

```
diff --git a/src/libhydra/attributes/mem_pool.c b/src/libhydra/attributes/mem_po
index f35ffaa9c244..0532f49e5379 100644
--- a/src/libhydra/attributes/mem_pool.c
+++ b/src/libhydra/attributes/mem_pool.c
@@ -376,7 +376,6 @@ METHOD(mem_pool_t, acquire_address, host_t*,
    switch (operation)
    {
        case MEM_POOL_EXISTING:
-           offset = get_existing(this, id, requested);
            break;
        case MEM_POOL_NEW:
            offset = get_new(this, id);
```

That way offline leases are only assigned if the IP pool is full, until then every client would get a new IP.

#2 - 05.03.2015 03:21 - richard hu

Sorry, I did not make my intent very clear.
get_existing is to check if the same id have an offline ip used before but did not used by other id yet.
what I care is get_reassigned. I want to avoid when id1 request ip, it will not be assigned by the ip that id2 just used and just release in last several seconds.
I saw we managed this by a hash_table leases, when get_reassigned it will look into the hash_table and find an old offline ip.
the hash table was sorted by the hash machanism and like a random sequence from user view, so sometime one ip was used by id1 and then used by id2 as soon as released by id1 in last second. this is what I want to avoid.
I read the code of mem_pool and make some change for my use.
I added a array_t to manage a FIFO sequence of all usable offline ips and every time request or release ip I will query this array_t.
there are some performance impact but if it is not outstanding I can accept it.
Please advise if I have any other issue need to be considered.

#3 - 05.03.2015 10:44 - Tobias Brunner

> the hash table was sorted by the hash machanism and like a random sequence from user view, so sometime one ip was used by id1 and then used by id2 as soon as released by id1 in last second. this is what I want to avoid.

So your goal is not to prevent this situation altogether (because if the pool is full you can't avoid reassigning IPs), but instead to increase the time between the release and reassignment of an IP (potentially to different clients)?

> I read the code of mem_pool and make some change for my use.

Please note that there will be some changes to that code with the upcoming 5.3.0 release (check the current master branch).

> I added a array_t to manage a FIFO sequence of all usable offline ips and every time request or release ip I will query this array_t.
> there are some performance impact but if it is not outstanding I can accept it.
> Please advise if I have any other issue need to be considered.

Hard to tell without seeing the code. Do you still use the offline arrays on entry_t objects? Or did you completely replace that with your global FIFO queue? If you didn't, you'd have to synchronize the lists of offline IPs somehow (with an impact on performance), otherwise you could end up assigning the same IP twice. Of course, if you don't use the offline arrays you can't directly reassign a lease to the same client, but have to consult the global queue. If that's not a problem for you I suppose such a global list of offline leases should work.

#4 - 09.03.2015 03:21 - richard hu

I did not have a chance to read master mem_pool code yet.
I add an extra FIFO queue(indeed a array_t) and keep the old hash table leases and according online and offline under every id and according machinisam code untouched.
As you said, not considering performance it should be ok because it has minimal impact for exist logic.
Do you have any performance idea for this? it add a blind go through for new FIFO queue in get_existing() and a blind go through for this->leases in get_reassigned()
Do you think this is a not accepted performance down for a production system with 4096 ip pool and 1200 conneted user and 30 connecting user.

If I get rid of above blind go through I need to delete the offline under this->leases. Then I need also delete get_existing().
What's your opinion if we do not use get_existing()? that means the client will get different ip even for a very short network disconnect and reconnect.

#5 - 09.03.2015 11:38 - Tobias Brunner

> Do you think this is a not accepted performance down for a production system with 4096 ip pool and 1200 conneted user and 30 connecting user.

You have to test that yourself (e.g. with load-tester). But 30 concurrent connection attempts is not that much and the list/table with 1200 connected users might also be small enough.

> If I get rid of above blind go through I need to delete the offline under this->leases. Then I need also delete get_existing().
> What's your opinion if we do not use get_existing()? that means the client will get different ip even for a very short network disconnect and reconnect.

Yes, without get_existing() your clients will always get a different IP. And to avoid syncing the lists you will have to get rid of that call (and the offline list on entry_t) otherwise you could end up assigning the same IP to different clients.

#6 - 10.03.2015 03:00 - richard hu

I know 30/1200 is not big number.
But the leases hash table was designed to contain all past user ids. not the current connected number.
So If you have enough new user, the hash table will grow as much as the vpn server boot up time.
If the vpn runs for 1 year without reboot, the hash table will contain many many ids.

Will it have any side effect if without get_existing()?

#7 - 11.03.2015 11:39 - Tobias Brunner

> But the leases hash table was designed to contain all past user ids. not the current connected number.
> So If you have enough new user, the hash table will grow as much as the vpn server boot up time.
> If the vpn runs for 1 year without reboot, the hash table will contain many many ids.

Yes, the table will fill with user IDs over time. I suppose we could modify get_reassigned a bit, so that entries without online or offline leases are

removed from the table (I did so in the *mem-pool-cleanup* branch). This should limit the number of entries in the table to at most the size of the address pool.

> Will it have any side effect if without get_existing()?

As mentioned, you will assign new IPs to all your clients (which is usually not a problem). In the current code get_exiting is actually also needed to reassign online leases in case of make-before-break reauthentication - so there you can't get rid of the whole function, but could strip out the offline lease handling. If you get rid of the offline array you could then remove entries from the hashtable in release_address when the last online lease was removed.

#8 - 07.07.2015 17:25 - Tobias Brunner

- Category set to libcharon

- Status changed from Feedback to Closed

- Assignee set to Tobias Brunner