strongSwan - Bug #1442

Memory leak when replacing existing CRLs (e.g. with ipsec rereadcrls)

28.04.2016 17:55 - nagi reddy

Status:	Closed	**Start date:**	28.04.2016
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	charon		
Target version:	5.5.0		
Affected version:	4.2.8	**Resolution:**	Fixed

Description

Hi,
when we are executing "/usr/local/strongswan/sbin/ipsec rereadcrl" command
1. Strongswan parses the crl file and populated its own data structures
2. If I execute the same command 2nd time with other different crl list, will strongswan free the old crl data structure list and populate new one? or it will append to old list?

we are using strongswan 4.2.8 version.

Awaiting for you valuable response.

Associated revisions

Revision 0ba905cf - 11.05.2016 12:16 - Tobias Brunner

mem-cred: Fix memory leak when replacing existing CRLs

Fixes #1442.

History

#1 - 28.04.2016 19:07 - Tobias Brunner

- Status changed from New to Feedback

> 2. If I execute the same command 2nd time with other different crl list, will strongswan free the old crl data structure list and populate new one? or it will append to old list?

Current strongSwan versions replace CRLs for the same issuer if they are newer (based on crlNumber or date). I'm not sure what you exactly mean with "append to old list". If you are referring to delta CRLs I don't think they are supported for static configuration (they are when fetching them from an URL though).

> we are using strongswan 4.2.8 version.

I've no idea what any program part did or did not do in 4.2.8.

#2 - 28.04.2016 19:17 - nagi reddy

for the first time if i have below crl list coming from the CR.
Serial Number: 0180E975
Revocation Date: Dec 18 13:05:14 2014 GMT
Serial Number: 0180DF9D
Revocation Date: Dec 18 13:05:14 2014 GMT
Serial Number: 0180CB29
Revocation Date: Dec 18 13:05:14 2014 GMT

second time i have below crl list coming from the CR.
Serial Number: 0180E975
Revocation Date: Dec 18 13:05:14 2014 GMT
Serial Number: 0180DF9F
Revocation Date: Dec 18 13:05:14 2014 GMT
Serial Number: 0180CB29
Revocation Date: Dec 18 13:05:14 2014 GMT

from above 2nd list only one is modified, So now in strongswan will have all 6 entries or only 4 entries or only 3 i.e from 2nd list?

#3 - 28.04.2016 19:47 - Tobias Brunner

> So now in strongswan will have all 6 entries or only 4 entries or only 3 i.e from 2nd list?

Only the entries of one of the CRLs, entries are not extracted. And as said which CRL is used depends on the serial numbers (crlNumber) or the dates of the CRLs.

#4 - 29.04.2016 16:19 - nagi reddy

when strongswan rereading crls using "/usr/local/strongswan/sbin/ipsec rereadcrls" command, charon memory keep on increasing, Do you need to run any command to free the memory from charon, before each rereadcrls command?

#5 - 29.04.2016 16:37 - Tobias Brunner

> when strongswan rereading crls using "/usr/local/strongswan/sbin/ipsec rereadcrls" command, charon memory keep on increasing, Do you need to run any command to free the memory from charon, before each rereadcrls command?

No idea. Maybe was a memory leak in 4.2.8 or perhaps it did just not replace loaded CRLs and simply added them to the internal list.

#6 - 30.04.2016 18:17 - nagi reddy

Even i tried with strongswan-5.4.0 version, still i am observing memory is keep on increasing. Even when i execute /usr/local/strongswan-5.4/sbin/ipsec purgecrls strongswan not clearing the crls, still i am able to see some info with /usr/local/strongswan-5.4/sbin/ipsec listcrls. Is this is memory leak in 5.4.0 version also? why the purgecrls is not clearing the crls from strongswan database?

#7 - 02.05.2016 14:19 - nagi reddy

Whenever rereadcrls request comes. function add() in x509_crl.c is called. Here parsing of crl file is done and private_x509_crl_t variable is created. This contatins crl pem file as well as revoked list.

Once we return from this function add_crl function in stroke_Cred.c is called from load_certdir function.

In this function current crl linkedlist is traveresd and compared with newely received crl file.
If there is difference between current crl file and newely received file, reference to older crl file is removed in following operation :-

```
this->certs->remove_at(this->certs, enumerator);
```

This node was containing pointer to older crl private_x509_crl_t variable.
We observed that while removing stroke list node, destroy function for private_x509_crl_t (x509_crl.c) is not called, there by resulted into memory leak (memory size allocated for private_x509_crl_t).

So final code for this is in function static bool add_crl() :-

```
if (found)
{
    new = cert->is_newer(cert, current);
    if (new)
{
    current->destroy(current);   /*Propsed fix to free memory*/

this->certs->remove_at(this->certs, enumerator);
    }
    else
{
    cert->destroy(cert);
    }
    break;
    }
```

In 5.2.0 add_Crl() is defined mem_cred.c.
In 4.2.8 add_crl() is defined in stroke_cred.c

The memory leak behaviour and code snipet is same in both versions.

Is our understanding and code change are correct?

#8 - 02.05.2016 16:09 - Tobias Brunner

- Tracker changed from Issue to Bug

- Subject changed from querey about rereadcrls to Memory leak when replacing existing CRLs (e.g. with ipsec rereadcrls)

- Assignee set to Tobias Brunner

- Target version set to 5.5.0

- Resolution set to Fixed

Thanks for testing the current version.

> Is our understanding and code change are correct?

Yes, you are absolutely correct. The leak was also in the *stroke* plugin before that code got moved to the mem_cred_t helper class (which happened with 4.5.1). I pushed a fix to the *1442-crl-replace-leak* branch of our repository.

#9 - 02.05.2016 16:21 - nagi reddy

Thanks for the quick response. we will continue with our testing.

#10 - 06.05.2016 07:58 - nagi reddy

After adding above fix to call destroy function to free memory, we are still seeing increase in charon heap memory.
1) We can see one check in destroy function for reference count.

x509_crl.c

```
static void destroy(private_x509_crl_t *this)
{
    if (ref_put(&this->ref))
    {
        this->revoked->destroy_function(this->revoked, free);
        DESTROY_IF(this->issuer);
        DESTROY_IF(this->authKeyIdentifier);
        free(this->encoding.ptr);
        free(this);
    }
}
```

we are not seeing above free() happening. we had added logs for same and verified.
so it means if reference count is greater than 1 it will not free the memory allocated for earlier CRL file.
What module or plugin could be using this memory, used for parsing of earlier CRL file?

#11 - 06.05.2016 09:22 - Tobias Brunner

> What module or plugin could be using this memory, used for parsing of earlier CRL file?

The CRL could still be in use somewhere, e.g. the certificate cache. Did you also run ipsec purcecrls? And how exactly do you measure the memory usage?

#12 - 06.05.2016 11:22 - nagi reddy

Thanks for response.
Following is block of configuration :-

cachecrls=no
charonstart=yes
plutostart=no
strictcrlpolicy=no

As per above configuration, cachecrl is no. since cachecrl is no, purgecrl is also not helping.
So I am not sure what are the other places apart from cache_cert() having access to crl references.
Can you please point out any hints or suspicious area in the code where we can isolate and study?

#13 - 06.05.2016 12:17 - Tobias Brunner

> As per above configuration, cachecrl is no. since cachecrl is no, purgecrl is also not helping.

No, that's not related. *cachecrl=yes* stores CRLs fetched from URIs in ipsec.d/crls. If a CRL (or any certificate really) is verified it is also cached in an in-memory certificate cache, which makes later verifications of the same certificate chains quicker (see source:src/libstrongswan/credentials/sets/cert_cache.c). The in-memory cache may be disabled with *charon.cert_cache=no* in strongswan.conf, or as mentioned flushed with purgecrls (which only affects the in-memory cache, the disk cache is not affected by it).

#14 - 09.05.2016 13:22 - nagi reddy

We were trying to understand purgecrl code in 5.4.0 version. We can see, we are clearing cert_cache_t data maintained as relation_t.
As per our understanding this list pointing to actual data structure.
For example if we have parsed one X509_CRL file than stroke plugin has one node pointing this.
One node in private_cert_cache_t, also pointing to same memory area in case of caching.
So when we are executing purgecrl command we are flushing node pointed in private_cert_cache_t and decreasing reference count.
we have not freed up stroke plugin node pointing towards actual allocated memory.
So that will be freed up only when new file will be parsed.
Is our understanding correct or purgecrl should clear up all references and actual memory too?

In case our understanding correct will there be any other place pointing same memory?
Second if there was difference between timing of purgecrl and rereadcrl will there be caching done before rereadcrl executed?

#15 - 09.05.2016 13:36 - Tobias Brunner

> Is our understanding correct or purgecrl should clear up all references and actual memory too?

No it does not clear all memory, it only removes CRL references in the certificate cache. As long as the CRL does not get replaced the credential set in the *stroke* plugin still has a reference to it.

> In case our understanding correct will there be any other place pointing same memory?

Could be, if e.g. an authentication is occurring at the moment. That's why refcounting is used.

> Second if there was difference between timing of purgecrl and rereadcrl will there be caching done before rereadcrl executed?

Possible as there could be threads using the CRL concurrently with you running either of these commands. You could run rereadcrl and perhaps slightly delayed purgecrls (does not really prevent this from happening, but chances are reduced).

#16 - 10.05.2016 09:13 - nagi reddy

We can see in cert_cache.c there is one cache size check.

static void check_cache(private_cert_cache_t *this) {

```
while (this->relations->get_count(this->relations) > CACHE_SIZE)
{
                    /*Loop Body to delete older one*/
                }
```

CACHE_SIZE is define as :-
#define CACHE_SIZE 30

So is this means that maximum cache size will be 30.

1) Is it means maximum number caching element is 30?

2) If we are ok with 30 cache elements than is there any need to do purge before or after reradcrl operation?

3) can we decrease size of caching fr eg: CACHE_SIZE = 10, will it create any other issue or affect any other leg?

#17 - 10.05.2016 10:47 - Tobias Brunner

> 1) Is it means maximum number caching element is 30?

Yes.

> 2) If we are ok with 30 cache elements than is there any need to do purge before or after reradcrl operation?

Yes, you'd probably want to run it after loading the new CRL. Because the old CRL will otherwise be served by the cache to the *revocation* plugin (as long as it is still valid).

> 3) can we decrease size of caching fr eg: CACHE_SIZE = 10, will it create any other issue or affect any other leg?

Sure, you can use whatever size you feel fits your scenario/PKI. The idea of the cache is to avoid expensive public key cryptography when verifying the same certificates multiple times.

#18 - 11.05.2016 11:17 - nagi reddy

When strongswan is starting up i can see we create caching for the loaded crl file.
Once new crl file reread occurred we clear stokr_list reference and create new node. But due to caching older one crl is stil in memory.
But I am not seeing any caching happening for the crl file loaded using rereadcrls.

1) We are trying to what are the scenarios in which caching would be triggered. Can you please point out some point here?

2) There is set method registerd while creating stroke plugin. How this set method hits? Is there some watcher kind of siting which will hit once new certificate comes or auth happens?

#19 - 11.05.2016 12:29 - Tobias Brunner

> Once new crl file reread occurred we clear stokr_list reference and create new node. But due to caching older one crl is stil in memory.

Again, that's why you'd call ipsec purgecrls after loading the new CRL.

> But I am not seeing any caching happening for the crl file loaded using rereadcrls.
>
> 1) We are trying to what are the scenarios in which caching would be triggered. Can you please point out some point here?

Whenever credential_manager_t::issued_by() is called. For example, when the validity of a certificate is checked this function is called for all certificates in the chain and if CRLs are checked by the *revocation* plugin then the CRL itself is verified.

Just loading a CRL should not immediately create a cache entry as the CRL's issuer is not verified at that point. In my tests just starting the daemon and loading CRLs (without initiating any connections) does not cache anything. Are you using *auto=start*?

> 2) There is set method registerd while creating stroke plugin. How this set method hits? Is there some watcher kind of siting which will hit once new certificate comes or auth happens?

What exactly are you referring to?

#20 - 12.05.2016 09:14 - nagi reddy

Thanks for explaining.
So until we execute purgecrl there will be two copy of crl file available in memory.
1) cached one
2) newer one read during rereadcrl.

In this case how charon takes decision, which one it has to use.
which leg of code will be doing this?

#21 - 12.05.2016 09:56 - Tobias Brunner

> So until we execute purgecrl there will be two copy of crl file available in memory.
> 1) cached one
> 2) newer one read during rereadcrl.

Yes.

> In this case how charon takes decision, which one it has to use.
> which leg of code will be doing this?

That's handled in the *revocation* plugin (source:src/libstrongswan/plugins/revocation/revocation_validator.c#L508). As long as the CRL in the cache is not stale it will get used (the certificates/CRLs from the cache are returned first by the certificate enumerator).

#22 - 30.05.2016 08:39 - nagi reddy

Thanks for the support, Memory leak is solved with the fix.

#23 - 30.05.2016 08:59 - Tobias Brunner

- Status changed from Feedback to Closed