strongSwan - Feature #3234

Intermediate certificates sent when using hash-and-url

29.10.2019 17:33 - Kyle Larose

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	libcharon		
Target version:	5.8.2		
Resolution:	Fixed		

Description

Hello,

We are using strongswan to connect to a third-party IPSEC VPN hosted by a customer. We are running into some issues with fragmentation of IKE messages, which
results in the VPN failing to establish. We thought that hash-and-url would be a reasonable solution to this problem, since the message size would be far under
our limit (~1400 bytes) if the certificates were not set.

However, we're now finding that in order to enable hash-and-url enabled, we must provide the intermediate certificate to strongswan (e.g. in cacerts). Now that
we've done that, it has started sending the intermediate certificate. While hash-and-url is successfully reducing the IKE message size by sending the end entity
certificate, strongswan has countered that by sending the intermediate certificate as well.

I found https://wiki.strongswan.org/issues/902 which discusses this problem. Tobias Brunner mentioned that at the time (about 4 years ago) strongswan did not
support sending only the end entity cert. This still appears to be the case as of 5.6.2 (ubuntu's build). The issue was closed due to lack of activity.

We need this functionality ASAP, so I am planning on building my own strongswan with it integrated. That said, if there are other ways in which we could
achieve not sending the intermediate certificates of which I am unaware, I'm all ears. Assuming I go ahead as planned, on want to contribute my changes
upstream, so any advice on the best approach would be helpful.

Some background on our situation:

- The customer's VPN appliance does not support IKEv2 fragmentation.
- The physical MTUs between the sites differ, which is why IP fragmentation is failing. We cannot reconcile them.
- We cannot sign our own certificates, and the CA we're using uses an intermediate certificate
- Running 5.6.2 on Ubuntu 18.04, and 5.7.2 on alpine 3.10.3

My suggested approach:
- New configuration parameter 'leftsendimcert'
- Two values: yes/no. Default yes.
- Modify **build_certs** (in ike_cert_post.ca) to only invoke **add_im_certs** if letsendimcert is yes.
- Not sure what to do about attribute certs. They're not a problem for us, I think.

Related issues:

Related to Issue #902: Intermediate certificate is sent when enable hash and ...	**Closed**

Associated revisions

Revision b290f43c - 26.11.2019 14:48 - Tobias Brunner

Merge branch 'hash-url-multi-level'

Adds support to send intermediate CA certificates in hash-and-URL
encoding. For that it moves the generation of URLs from the config
backends to the ike-cert-post task.

Fixes #3234.

History

#1 - 30.10.2019 11:37 - Tobias Brunner

- Status changed from New to Feedback

> I found https://wiki.strongswan.org/issues/902 which discusses this problem. Tobias Brunner mentioned that at the time (about 4 years ago) strongswan did not
> support sending only the end entity cert. This still appears to be the case as of 5.6.2 (ubuntu's build).

Yes, nothing has changed in that regard. I think it wouldn't be too difficult to add support for sending intermediate CA certs as hash-and-URL (provided the base URI for the parent CA has been configured).

> That said, if there are other ways in which we could
> achieve not sending the intermediate certificates of which I am unaware, I'm all ears.

You could obviously just configure the end-entity certificate of the peer to establish trust (no root/intermediate CA required). If you do that on both ends, you probably don't have to send any certificates at all (*left|rightsendcert=never*).

> - The customer's VPN appliance does not support IKEv2 fragmentation.

Try lobbying for an upgrade or an alternative solution that supports IKEv2 fragmentation (although you might later have issues with fragmented ESP packets anyway if you don't fix your MTU/MSS).

> - The physical MTUs between the sites differ, which is why IP fragmentation is failing. We cannot reconcile them.

Fix PMTUD or use a low enough, manually configured MTU on both ends.

> - We cannot sign our own certificates, and the CA we're using uses an intermediate certificate

Why can't you use your own certificates?

> - New configuration parameter 'leftsendimcert'

There won't be any new config options for the legacy ipsec.conf. If anything, the option would be added to swanctl.conf/vici.

> - Modify **build_certs** (in ike_cert_post.ca) to only invoke **add_im_certs** if letsendimcert is yes.

Maybe, however, that seems like a rare corner use case.

#2 - 30.10.2019 11:38 - Tobias Brunner

- Related to Issue #902: Intermediate certificate is sent when enable hash and url and configured CA section with certuribase added

#3 - 30.10.2019 13:47 - Kyle Larose

Thanks for the quick feedback.

Tobias Brunner wrote:

> > I found https://wiki.strongswan.org/issues/902 which discusses this problem. Tobias Brunner mentioned that at the time (about 4 years ago) strongswan did not
> > support sending only the end entity cert. This still appears to be the case as of 5.6.2 (ubuntu's build).
>
> Yes, nothing has changed in that regard. I think it wouldn't be too difficult to add support for sending intermediate CA certs as hash-and-URL (provided the base URI for the parent CA has been configured).

I didn't even consider that option! It's much more elegant than what I proposed. For some reason I had convinced myself early on that system would only support dealing with the end entity cert, but that doesn't really make much sense.

I've implemented my earlier proposed change, but I prefer your proposal. I'll try that out. If it works, then I'll discard my original proposal and go with yours.

> You could obviously just configure the end-entity certificate of the peer to establish trust (no root/intermediate CA required). If you do that on both ends, you probably don't have to send any certificates at all (*left|rightsendcert=never*).

This is certainly an option, though we like to rotate our certificates regularly, so it may be challenging to manage.

> > - The customer's VPN appliance does not support IKEv2 fragmentation.
>
> Fix PMTUD or use a low enough, manually configured MTU on both ends.

Unfortunately their VPN appliance doesn't seem to have any options that would help us here with IKE. For ESP, the device uses a tunnel interface on which you may configure an MTU. But, it performs IKE over its physical device. Devices outside of our control are connecting to it: we can't change the MTU there without affecting them.

> > - We cannot sign our own certificates, and the CA we're using uses an intermediate certificate
>
> Why can't you use your own certificates?

This is the PKI we're using. We don't want to special case our VPN.

> > \- New configuration parameter 'leftsendimcert'
>
> There won't be any new config options for the legacy ipsec.conf. If anything, the option would be added to swanctl.conf/vici.

I **think** I updated these in the work I did yesterday (I pretty much applied the change to anywhere (left|right)sendcert existed). I'm not sure how to test them,
though. If I decided to proceed with this, I'll make sure to.

> > \- Modify **build_certs** (in ike_cert_post.ca) to only invoke **add_im_certs** if letsendimcert is yes.
>
> Maybe, however, that seems like a rare corner use case.

#4 - 30.10.2019 14:42 - Tobias Brunner

- File intermediate-hash-url.patch added

> > Yes, nothing has changed in that regard. I think it wouldn't be too difficult to add support for sending intermediate CA certs as hash-and-URL (provided the base URI for the parent CA has been configured).
>
> I didn't even consider that option! It's much more elegant than what I proposed. For some reason I had convinced myself early on that system would only support dealing with the end entity cert, but that doesn't really make much sense.

The attached patch might already be enough (only compile tested), at least when loading certs from default directories (additional changes will probably be necessary to make this work with *ca/authority* sections, in particular because of how certs are currently associated with such sections for hash-and-URL use).

#5 - 30.10.2019 16:15 - Kyle Larose

- File loop.patch added

Tobias Brunner wrote:

> The attached patch might already be enough (only compile tested), at least when loading certs from default directories (additional changes will probably be necessary to make this work with *ca/authority* sections, in particular because of how certs are currently associated with such sections for hash-and-URL use).

Thanks! This definitely sent me on the right path. I got the **build_cert_payload** bit, but had no idea how to make it actually work. However, the patch as-is
did not work for me. I appear to have got it working, but I don't think it's right. I'll elaborate.

1. I think we have a double negative on the **issued_by** call -- we're checking for any certificate that isn't issued by itself, right?
2. More importantly, there is a dependency problem. When **load_x509_ca** is called, the **stroke_ca_t** has not populated its section list yet. This means

that
check_for_hash_and_url will not be able to find the parent certificate. I tried adding the check to **private_stroke_ca_t::add**, but there is another issue whereby if the issuer for the intermediate has not been added yet, the search will again fail.

My hackish solution to deal with number 2 was to simply iterate over every ca on each call to add, and invoke **check_for_hash_and_url**. It worked. But, I'm not a fan for a few reasons: it seems inefficient (why not just do it once at the end), and without some refactoring, I think the locking may be messed up.

I'm not sure whether this problem applies to the vici portion of the patch. I'm also not sure whether what I've done will work on reload/replace.

Some other options:

1. Create a new public function: stroke_ca_t::update_ca_hash_and_url. Invoke it once all the ca certs have been added (where would we do that)? Invoke the private side of it on reload/replace.
2. Figure out how to control the order in which certs are added so that users can define them in least-dependency order.

The first option seems the least error prone from an end-user perspective.

I've attached my hacked up version of your patch. Note: this is not production ready; I'm not locking on the enumeration, and I suspect I'm supposed to.

Do you have any thoughts on how to proceed from here?

#6 - 30.10.2019 17:34 - Tobias Brunner

> 1. I think we have a double negative on the **issued_by** call -- we're checking for any certificate that isn't issued by itself, right?

Yep, typo, should only be one ! there.

> 2. More importantly, there is a dependency problem. When **load_x509_ca** is called, the **stroke_ca_t** has not populated its section list yet. This means that
> **check_for_hash_and_url** will not be able to find the parent certificate. I tried adding the check to **private_stroke_ca_t::add**, but there is another issue whereby if the issuer for the intermediate has not been added yet, the search will again fail.

Yes, as mentioned, my patch doesn't work with *ca* sections. However, the intermediate CA certificate, if stored in ipsec.d/cacerts, should be loaded separately and associated with an already loaded *ca/authority* section.

> My hackish solution to deal with number 2 was to simply iterate over every ca on each call to add, and invoke **check_for_hash_and_url**. It worked. But, I'm not a fan for a few reasons: it seems inefficient (why not just do it once at the end), and without some refactoring, I think the locking may be messed up.

Well, it's not called that often as there are usually very few *ca/authority* sections and rarely any updates. So this might be an OK temporary solution.

A better approach would probably be to not cache the hashes at all, but pass the certificate directly to the credential sets (as create_cdp_enumerator takes an identity, i.e. identification_t*, it would be kind of a hack to directly do so, but since it's just a pointer it would probably work without much issues, at least if all sets check the type and ignore CERT_X509). Making the function take varargs that depend on the type would be a relatively safe option to that, I guess (but requires more changes). Similarly, using a separate enumerator for such URIs would be an option but requires several more changes (credential manager and sets).

> I'm not sure whether this problem applies to the vici portion of the patch.

No, that needs some work too.

> I'm also not sure whether what I've done will work on reload/replace.

In *stroke* it should work fine, as starter just deletes and (re-)adds stuff (however that might leave hashes in the list).

> 1. Create a new public function: stroke_ca_t::update_ca_hash_and_url. Invoke it once all the ca certs have been added (where would we do that)? Invoke the private side of it on reload/replace.

It's not known when all these certs/sections have been loaded, actually, they could even be loaded by a different backend. So doing this dynamically would be the most flexible approach.

> 2. Figure out how to control the order in which certs are added so that users can define them in least-dependency order.

Can't really be controlled.

#7 - 30.10.2019 19:14 - Tobias Brunner

- Tracker changed from Issue to Feature

- Affected version deleted (5.6.2)

OK, found an even simpler solution: Just enumerate the base URI that is registered for a specific issuer identity, then build the URL directly in ike_cert_post.c. I pushed a quick WIP commit to the *3234-hash-url* branch.

#8 - 30.10.2019 22:56 - Kyle Larose

Tobias Brunner wrote:

> OK, found an even simpler solution: Just enumerate the base URI that is registered for a specific issuer identity, then build the URL directly in ike_cert_post.c. I pushed a quick WIP commit to the *3234-hash-url* branch.

Awesome. I just tried it out, and it worked. It was a very simple test, but it sent both the hash and urls, which is my pass criteria. :)

#9 - 31.10.2019 15:27 - Tobias Brunner

- Category set to libcharon

- Assignee set to Tobias Brunner

- Target version set to 5.8.2

> Awesome. I just tried it out, and it worked. It was a very simple test, but it sent both the hash and urls, which is my pass criteria. :)

Great, thanks for testing. I updated the branch (proper commits etc. and also added a test scenario).

#10 - 01.11.2019 18:09 - Kyle Larose

Tobias Brunner wrote:

> > Awesome. I just tried it out, and it worked. It was a very simple test, but it sent both the hash and urls, which is my pass criteria. :)
>
> Great, thanks for testing. I updated the branch (proper commits etc. and also added a test scenario).

Latest patch works as well. Thanks a lot for doing this.

#11 - 26.11.2019 14:50 - Tobias Brunner

- Status changed from Feedback to Closed

- Resolution set to Fixed

Files

intermediate-hash-url.patch	1.87 KB	30.10.2019	Tobias Brunner
loop.patch	1.94 KB	30.10.2019	Kyle Larose