strongSwan - Feature #2451

Adding functionality for RADIUS relay of Class AVP to accounting server

18.10.2017 17:00 - Thomas Strangert

Status:	Closed	**Start date:**	18.10.2017
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	libcharon		
Target version:	5.6.1		
Resolution:	Fixed		

Description

Added an optional feature to the eap-radius plugin:
"eap-radius.accounting_relay_class" (default no)

If enabled, then the first **Class** AVP in incoming RADIUS *Access-Accept* messages will be relayed to the accounting server as a part of consequential *Accounting-Request* start/interim/stop messages. (Ref: RFC 2865 section 5.25)

The idea/usage of this functionality is that a RADIUS server granting access to strongSwan can generate and send along a unique Class AVP that will be relayed by the RADIUS client (e.g. strongSwan) to the accaunting server, hence creating an easy way on the RADIUS server side to uniquely correlate authorisation sessions with accounting sessions.

Note: Requires the *eap-radius.accounting* option enabled.

Associated revisions

Revision 65592407 - 02.11.2017 09:57 - Tobias Brunner

eap-radius: Optionally send Class attributes in RADIUS accounting messages

If enabled, add the RADIUS Class attributes received in Access-Accept messages
to RADIUS accounting messages as suggested by RFC 2865 section 5.25.

Fixes #2451.

History

#1 - 18.10.2017 18:33 - Tobias Brunner

- Status changed from New to Feedback

Thanks for the suggestion and patch.

I suppose we could use the existing *eap-radius.class_group* functionality for this. But I guess technically this use of the attribute is rather the opposite (i.e. not a group identifier but a unique client identifier). So that might be confusing.

But I'd still prefer to use parts of that functionality for this instead of duplicating the parsing and adding new members to the ike_sa_t class. I did that in the *2451-radius-accounting-class* branch. The conversion to identification_t might not be ideal in case the attribute does not contain a string. So that might have to be changed to chunk_t* (requiring some work in auth_cfg_t as there are currently no rules that store chunk_t pointers) if it turns out to be a problem. Since all received attributes are stored we could also send all of them, but I guess there usually won't be more than one?

Not sure if making this configurable is required (or is there a downside to forwarding the attribute, besides the increased message size?).

#2 - 19.10.2017 18:12 - Thomas Strangert

Thank you for your fast response and feedback :)

I've had a look at the code in the new branch you created and compared it with what I submitted. With that, plus your comments above, I have the following comments:

- My code / patch aimed to have minimal impact on existing code and to be functionally separated/backwards compatible and/or simply because I figured that you would prefer to have "atomic" additions. For instance, I avoided to interleave my stuff into the existing process_class(), instead creating a similar process_class_relay() function that kicked in only when my proposed option switch *eap-radius.accounting_relay_class* was enabled.

- I wasn't fully aware of the auth_cfg_t functionality (in particular the usage of get_auth_cfg() and add() methods) and what implications it would have if I started to add() my stuff with it. This was why I chose to create a variable and a set()/get() to the ike_sa_t class instead. As long as auth_cfg_t keeps its data/has the same lifespan as ike_sa_t then I have no opinion on where to store the Class AVPs.

- Regarding the lifespan/how to keep and update the Class AVPs: It is important that the Class AVPs sent out to the accounting server are the same as the ones received in the **most recent** Access-Accept message (*and also in the **same order**!*). Theoretically, an Access-Accept message could be a follow-up re-authentication with updated Class data or even *without* Class AVPs, in either case consequent Accounting-Request messages shouldn't continue relaying initial/old Class data.

- I would strongly advice to have my proposed separate option switch eap-radius.accounting_relay_class (*default no*) enabling the relay function and not to bundle it with the existing eap-radius.class_group. There might be accounting servers in service out there that have support for incoming Class relay, but presently doesn't get that stimuli from strongSwan clients and there's no telling how those would react if they suddenly do! Also, I am not aware of how the class_group functionality as it exists in strongSwan will react if the RADIUS authorisation server starts to pump seemingly random, binary data in Class AVPs, that are really meant only to be interpreted in an accounting server?

- Hence: for the intents and purposes of RFC 2865 section 5.25, the Class AVP data contains 1-253 arbitrary binary octets, so no C-code string handling implied or guaranteed. chunk_t/memcpy() handling a must!

#3 - 23.10.2017 11:35 - Tobias Brunner

- My code / patch aimed to have minimal impact on existing code and to be functionally separated/backwards compatible and/or simply because I figured that you would prefer to have "atomic" additions. For instance, I avoided to interleave my stuff into the existing process_class(), instead creating a similar process_class_relay() function that kicked in only when my proposed option switch *eap-radius.accounting_relay_class* was enabled.

It always depends :) But the existing code in process_class() already does exactly the same thing you wanted (store the received attribute(s) somewhere).

- I wasn't fully aware of the auth_cfg_t functionality (in particular the usage of get_auth_cfg() and add() methods) and what implications it would have if I started to add() my stuff with it. This was why I chose to create a variable and a set()/get() to the ike_sa_t class instead. As long as auth_cfg_t keeps its data/has the same lifespan as ike_sa_t then I have no opinion on where to store the Class AVPs.

I try to avoid adding new members to major classes, in particular for features only a minority of users will use. And, yes, the lifespan of the auth data is the same as that of the IKE_SAs (unless the option charon.flush_auth_cfg is enabled). Another option would be to store them in the eap-radius plugin (in a hashtable mapping IKE_SAs to Class attributes).

- Regarding the lifespan/how to keep and update the Class AVPs: It is important that the Class AVPs sent out to the accounting server are the same as the ones received in the **most recent** Access-Accept message (*and also in the **same order**!*). Theoretically, an Access-Accept message could be a follow-up re-authentication with updated Class data or even *without* Class AVPs, in either case consequent Accounting-Request messages shouldn't continue relaying initial/old Class data.

I don't think that's an issue here. There is no EAP/RADIUS reauthentication with IKEv2 (a new IKE_SA has to be created to reauthenticate). So there won't be any update of these attributes. The only thing that could happen is using multiple authentication rounds, but using EAP for more than one is very uncommon (if at all possible). The order of rules in auth_cfg_t is maintained when enumerating them (however, only the first attribute is currently used).

- I would strongly advice to have my proposed separate option switch eap-radius.accounting_relay_class (*default no*) enabling the relay function and not to bundle it with the existing eap-radius.class_group. There might be accounting servers in service out there that have support for incoming Class relay, but presently doesn't get that stimuli from strongSwan clients and there's no telling how those would react if they suddenly do! Also, I am not aware of how the class_group functionality as it exists in strongSwan will react if the RADIUS authorisation server starts to pump seemingly random, binary data in Class AVPs, that are really meant only to be interpreted in an accounting server?

It's not bundled with that other option at all. But sure, we can add a separate option to enable this.

- Hence: for the intents and purposes of RFC 2865 section 5.25, the Class AVP data contains 1-253 arbitrary binary octets, so no C-code string handling implied or guaranteed. chunk_t/memcpy() handling a must!

Well, that RFC is 17 years old. So do people currently use random binary data here, or are these generally hex/base64 strings?

#4 - 23.10.2017 14:19 - Tobias Brunner

I pushed a version that stores the attributes in the *eap-radius* plugin (and has an option to enable it) to the *2451-radius-accounting-class-ht* branch.

#5 - 24.10.2017 11:43 - Thomas Strangert

Hi,

Built and tested (to a limited extent) the 2451-radius-accounting-class-ht branch - **OK!**
Did not test sending multiple Class to see if those were relayed (and also in order).

I'm happy with the patch, you may merge the branch to master/HEAD now.

#6 - 02.11.2017 09:58 - Tobias Brunner

- Status changed from Feedback to Closed

- Assignee set to Tobias Brunner

- Target version set to 5.6.1

- Resolution set to Fixed

Thanks for testing. Merged to master.

Files

0001-Adding-functionality-for-RADIUS-relay-of-Class-AVP-t.patch	11.2 KB	18.10.2017	Thomas Strangert
relay-class-attribute.pcapng	13.7 KB	18.10.2017	Thomas Strangert
ipsec.log	142 KB	18.10.2017	Thomas Strangert
relay-class-attribute-source-files.zip	41 KB	18.10.2017	Thomas Strangert