strongSwan - Bug #3284

OpenSSL 1.1.1c on RHEL-8/Fedora 29+ breaks eap-radius and some tunnels

02.12.2019 17:34 - Assen Totin

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	libstrongswan		
Target version:	5.8.2		
Affected version:	5.7.2	**Resolution:**	Fixed

Description

Hello!

I have StrongSwan 5.7.2 (EPEL) on RHEL-8 using the OS-provided OpenSSL 1.1.1. After OpenSSL got upgraded to 1.1.1c (RHEL-8.1), the EAP-RADIUS module of StrongSwan stopped working, saying either HASH_MD5 or RNG_WEAK was not available in libcrypto.so. As a result, eap-radius module refused to load and log message was "RADIUS initialization failed, HMAC/MD5/RNG required", comming from https://github.com/strongswan/strongswan/blob/57447015db828832e0e141dcdab7fbf61f828851/src/libradius/radius_socket.c around like 410.

Our use case: we need Windows/Mac/Linux/Android/iOS clients to connect over VPN using their Active Directory accounts without the need to install third-party software. We employ IPSec for this with Strongswan -> FreeRADIUS -> MSCHAPv2 -> Windows AD (hence we need StrongSwan as the RHEL-provided LibreSwan is uncapable of this whatsoever).

In addition, some net-2-net tunnels refused to start after the OpenSSL upgrade; I could not check them before we downgraded OpenSSL, but I'd assume it affected tunnels where we use 3des-sha1-modp1024.

This bug report from months ago explains a similar issue, though related to MD5 - which got zero attention from RedHat: https://bugzilla.redhat.com/show_bug.cgi?id=1741867

Downgrading OpenSSL to original 1.1.1 on RHEL-8 fixed the issue.

Is this something known or observed/reported before? Is there a recommended way of dealing with this? Keeping an older OpenSSL around for prolonged period is not seen as a good thing. Any patch/update of StrongSwan that may resolve this, or any other ideas?

Thank you in advance,

Assen Totin

Associated revisions

Revision 6b347d52 - 06.12.2019 10:27 - Tobias Brunner

openssl: Ensure underlying hash algorithm is available during HMAC init

Without this we only would learn that the algorithm isn't actually
available (e.g. due to FIPS mode) when set_key() is called later, so there
isn't any automatic fallback to other implementations.

Fixes #3284.

History

#1 - 02.12.2019 19:13 - Tobias Brunner

- Status changed from New to Feedback

> As a result, eap-radius module refused to load and log message was "RADIUS initialization failed, HMAC/MD5/RNG required"

Yep, without MD5 you can't use RADIUS, which is also why you can't use it with OpenSSL in FIPS mode (#1217).

> I could not check them before we downgraded OpenSSL, but I'd assume it affected tunnels where we use 3des-sha1-modp1024.

Hm, not sure what could fail there.

> Is this something known or observed/reported before?

Not that I'm aware.

> Is there a recommended way of dealing with this?

Nope, sorry.

> Any patch/update of StrongSwan that may resolve this, or any other ideas?

Nope.

#2 - 02.12.2019 19:27 - Assen Totin

For the record, I know about FIPS and we are not in FIPS mode on RHEL.

I guess I was one of the first bitten as we run real RHEL-8 - while most StrongSwan users of the RedHat ecosystem are, probably, on CentOS, which will become 8.1 (and get this OpenSSL update) in a few weeks.

Looking into the StrongSwan code, it seems it uses a lot of custom wrappers, like create_hasher(), create_signer() and create_rng() for example; would it be possible for somebody who knows the code to distil this done to a simple few lines of actual libcrypto.so calls, so that we may at least figure out which of the three is broken - HASH_MD5 (is this HMAC MD5?), AUTH_HMAC_MD5_128 or RNG_WEAK? In the light of the quoted ticket above, I strongly suspect MD5, but would really like to know before bisecting code snapshots - the CLI of OpenSSL does not reveal anything immediately obvious in its regard.

#3 - 03.12.2019 12:09 - Tobias Brunner

> For the record, I know about FIPS and we are not in FIPS mode on RHEL.

OpenSSL 1.1.1 isn't compatible to the OpenSSL FIPS module anyway (it can only be used with OpenSSL 1.0.x).

> so that we may at least figure out which of the three is broken - HASH_MD5 (is this HMAC MD5?), AUTH_HMAC_MD5_128 or RNG_WEAK?

Your first check should be the loaded algorithms (ipsec listalgs or swanctl --list-algs, note that any stronger RNG is accepted for RNG_WEAK). If you see all of them, the instantiation apparently fails.

One issue is that if the *openssl* plugin isn't built against the headers of your actually used OpenSSL version there might be algorithms registered that are not actually supported by the underlying library (e.g. if OpenSSL was built without MD5 support OPENSSL_NO_MD5 would be in the config header, so that algorithm wouldn't be registered by the plugin; but if that wasn't the case in the version against which the plugin in the EPEL package was built, MD5 is still registered, but instantiation will later fail).

HASH_MD5 is basically instantiated with something like this:

```
const EVP_MD *hasher = EVP_get_digestbyname("md5");
if (hasher)
{
    EVP_MD_CTX *ctx = EVP_MD_CTX_create();
    if (EVP_DigestInit_ex(ctx, hasher, NULL) == 1)
    {
        /* success */
    }
}
/* failure */
```

very similar for AUTH_HMAC_MD5_128:

```
const EVP_MD *hasher = EVP_get_digestbyname("md5");
if (hasher)
{
    HMAC_CTX *ctx = HMAC_CTX_new();
    /* this is what the set_key() method does: */
    char *key = "secretkey";
    if (HMAC_Init_ex(ctx, key, strlen(key), hasher, NULL))
    {
```

```
        /* success */
    }
}
/* failure */
```

#4 - 03.12.2019 13:57 - Assen Totin

Thank you for the code samples. Fortunately (or not?) both of them give "success" - even when compiled against OpenSSL-1.1.1 (RHEL-8.0) but run against OpenSSL-1.1.1c (RHEL-8.1).

I managed to narrow down the issue to the following:

```
OpenSSL-1.1.1 (RHEL-8.0)
     [root@cgvpnasa assen.totin]# swanctl --list-algs | grep HMAC_MD5_128
       HMAC_MD5_128[hmac]

OpenSSL-1.1.1c (RHEL-8.1, same StrongSwan binaries/libraries as above)
     [root@cgvpnasa assen.totin]# swanctl --list-algs | grep HMAC_MD5_128
       HMAC_MD5_128[openssl]
```

In my case both openssl.conf and hmac.conf have "load = yes", so they should be same priority. How does in this case StrongSwan decide which module to use for a given crypto function? I.e. could it be that OpenSSL-1.1.1c *added* something instead of removing it, hence it became the provider of chocie for HMAC_MD5?

Additionally, with OpenSSL-1.1.1c (RHEL-8.1) I get this line in charon.log, which I do not have with OpenSSL-1.1.1:

```
[LIB] openssl FIPS mode(2) - enabled
```

FIPS is not enabled globally at OS level. Knowing that OpenSSL 1.1.1 is not supposed to have FIPS... what does this mean? RHEL have added patches that still enforce FIPS on 1.1.1c somehow?

My current workaround is to set "load = 10" in hmac.conf, which makes it the provider of choice for HMAC_MD5 and eap-radius loads again.

But the question remains, since HMAC_Init_ex() succeeds on OpenSSL-1.1.1c (as per your sample code above, run on the same VM as StrongSwan), why or what does fail for StrongSwan then?

#5 - 03.12.2019 16:25 - Tobias Brunner

> How does in this case StrongSwan decide which module to use for a given crypto function?

With equal priorities the order is determined by the default plugin list (as created by the configure script, see source:configure.ac#L1365), which has *openssl* listed before *hmac*. So if you actually see that the *hmac* plugin provides the algorithm, it means the *openssl* plugin isn't loaded or it doesn't register the algorithm (which algorithms are registered is determined at compile time based on the constants I mentioned before), or, if you have the crypto tests/benchmarking enabled "on add", that the implementation was demoted or removed.

> I.e. could it be that OpenSSL-1.1.1c *added* something instead of removing it, hence it became the provider of chocie for HMAC_MD5?

Seems strange if you use the same version of the *openssl* plugin, unless it is reordered or not loaded successfully at all. So check the complete list of algorithms and perhaps the log for details.

> Additionally, with OpenSSL-1.1.1c (RHEL-8.1) I get this line in charon.log, which I do not have with OpenSSL-1.1.1:
>
> [LIB] openssl FIPS mode(2) - enabled
>
> FIPS is not enabled globally at OS level.

That would at least explain why MD5 fails.

> Knowing that OpenSSL 1.1.1 is not supposed to have FIPS... what does this mean? RHEL have added patches that still enforce FIPS on 1.1.1c somehow?

No idea. Perhaps there are two versions of the library around (which would be problematic as OpenSSL 1.0 is quite different from OpenSSL 1.1, so building against one and then linking the other at runtime wouldn't work well). Or they actually have FIPS-certified their version of OpenSSL 1.1. Did you try disabling FIPS mode via *charon.plugins.openssl.fips_mode*?

> My current workaround is to set "load = 10" in hmac.conf, which makes it the provider of choice for HMAC_MD5 and eap-radius loads again.

Interesting, because the *hmac* plugin still requires an HASH_MD5 implementation. Which plugin provides that in your case? If it's the *openssl* plugin

it's weird that that would work if FIPS mode is enabled (unless you have other plugins loaded that provide it so there is a fallback).

Theoretically, the order shouldn't really matter because even if instantiation fails for one plugin, the crypto factory will fall back to other implementations. However, the problem with the *openssl* plugin's current HMAC implementation is that the instantiation doesn't do very much, so that might very well succeed. It's calling set_key() afterwards that actually initializes the HMAC_CTX instance with HMAC_Init_ex(), which will fail if MD5 is not available due to FIPS mode. I guess that's what happens here.

I pushed a commit to the *3284-openssl-hmac* branch, which initializes the OpenSSL HMAC_CTX instance with an empty key during instantiation. This should make sure we know whether the underlying hash algorithm is available. If not, the crypto factory can automatically fall back to e.g. the *hmac* plugin if it is loaded, no late failures when setting the key.

#6 - 03.12.2019 22:33 - Assen Totin

Thank you, looks like we're getting close on this.

> Did you try disabling FIPS mode via charon.plugins.openssl.fips_mode?

No, because I saw the default value there was "0", i.e. disabled. However, when I actively set it to "0" today, this fixed the issue on unpatched StrongSwan-5.7.2.

> Interesting, because the *hmac* plugin still requires an HASH_MD5 implementation. Which plugin provides that in your case? If it's the *openssl* plugin it's weird that that would work if FIPS mode is enabled (unless you have other plugins loaded that provide it so there is a fallback).

On both systems, OpenSSL-1.1.1 and OpenSSL-1.1.1c MD5 is provided by the "md5" StrongSwan module (some built-in implementation, perhaps?) - this is why HMAC-MD5 from "hmac" works. On both systems "md5" is higher on the list of loaded plugins than "openssl", which explains why MD5 is taken from it.

Also on both systems "hmac" is below "openssl" on the list - so I'd guess that with the earlier OpenSSL-1.1.1 the "openssl" is not chosen for HMAC_MD5 at start-up time for some reason and "hmac" is used instead. On the newer OpenSSL-1.1.1c, however, "openssl" gets chosen and then fails at runtime. Build against OpenSSL-1.1.1 and OpenSSL-1.1.1c behave the same way.

Your patch from 3284-openssl-hmac branch also works fine - when applied to StrongSwan 5.7.2 with OpenSSL-1.1.1c (without touching the *fips_mode* of openssl module) the eap-radius works. The "openssl" module is still before "hmac" on the list and HMAC_MD5_128 is still listed as being provided by "openssl" swanctl --list-algs, but the failover works.

Maybe I should send this patch to the StrongSwan maintainer in EPEL for inclusion in the current version, 5.7.2? And/or have a note on your wiki that, starting with RHEL-8.1, it is necessary to explicitly enable *fips_mode=0* due to the way they build OpenSSL from 1.1.1c onward?

#7 - 04.12.2019 11:21 - Tobias Brunner

- Tracker changed from Issue to Bug

- Category set to libstrongswan

- Target version set to 5.8.2

> > Did you try disabling FIPS mode via charon.plugins.openssl.fips_mode?
>
> No, because I saw the default value there was "0", i.e. disabled. However, when I actively set it to "0" today, this fixed the issue on unpatched StrongSwan-5.7.2.

The default can be changed with the --with-fips-mode configure option, which the maintainers of the EPEL package do.

> On both systems, OpenSSL-1.1.1 and OpenSSL-1.1.1c MD5 is provided by the "md5" StrongSwan module (some built-in implementation, perhaps?) - this is why HMAC-MD5 from "hmac" works. On both systems "md5" is higher on the list of loaded plugins than "openssl", which explains why MD5 is taken from it.

I see. Yes, that's our own implementation in the *md5* plugin (in non-FIPS mode it's redundant).

> Also on both systems "hmac" is below "openssl" on the list - so I'd guess that with the earlier OpenSSL-1.1.1 the "openssl" is not chosen for HMAC_MD5 at start-up time for some reason and "hmac" is used instead. On the newer OpenSSL-1.1.1c, however, "openssl" gets chosen and then fails at runtime.

Not sure why that would be the case (except for the reasons I mentioned before: crypto tests/benchmarking/load error).

> Your patch from 3284-openssl-hmac branch also works fine - when applied to StrongSwan 5.7.2 with OpenSSL-1.1.1c (without touching the *fips_mode* of openssl module) the eap-radius works.

Great, thanks for testing.

> Maybe I should send this patch to the StrongSwan maintainer in EPEL for inclusion in the current version, 5.7.2?

Sure, go ahead.

> And/or have a note on your wiki that, starting with RHEL-8.1, it is necessary to explicitly enable *fips_mode=0* due to the way they build OpenSSL from 1.1.1c onward?

Hopefully, this issue will come up when someone searches for a similar issue.

It's still interesting that RHEL 8 actually has a FIPS-certified version of OpenSSL 1.1 (unfortunately, this article on the RH site doesn't mention the OpenSSL version or other details, and this article has not yet been updated for RHEL 8).

#8 - 04.12.2019 12:26 - Assen Totin

> The default can be changed with the --with-fips-mode configure option, which the maintainers of the EPEL package do.

Aggggh... oddly, it was also set to mode "2" in 5.7.2 (and many previous versions) which we used with OpenSSL-1.1.1; it is also set to 2 for RHEL-7 which comes with OpenSSL-1.0 series... and in neither case this caused problems with MD5.

This means something changed very recently, either in OpenSSL-1.1.1c or in the way RedHat ships it. As per https://wiki.openssl.org/index.php/FIPS_mode_set():

> Currently all non-zero values of ONOFF enable FIPS mode. In the future other values may specify additional actions beyond enabling FIPS mode, such as a value of 2 to designate an additional restriction to Suite B algorithms.

Maybe this finally became true somehow. It is still puzzling why such an important change happened in a minor RHEL release and without a single word in their Release Notes.

I have a ticket opened with RedHat support, if they ever say anything, I'll update this ticket to keep it for posterity.

This was a very helpful (and learning, albeit somewhat stressful) experience - thank you very much for your help! You may close this issue, I guess.

#9 - 04.12.2019 18:48 - Tobias Brunner

> Aggggh... oddly, it was also set to mode "2" in 5.7.2 (and many previous versions) which we used with OpenSSL-1.1.1; it is also set to 2 for RHEL-7 which comes with OpenSSL-1.0 series... and in neither case this caused problems with MD5.

Yeah, but I guess the same thing happens there as does on RHEL 8 with 1.1.1, i.e. HMAC_MD5_128 is provided by the *hmac* plugin by default (why is still unclear to me).

> This means something changed very recently, either in OpenSSL-1.1.1c or in the way RedHat ships it.

Again, what's really strange is the change from *hmac* to *openssl* for HMAC_MD5_128 just by changing the library.

> As per https://wiki.openssl.org/index.php/FIPS_mode_set():
>
> > Currently all non-zero values of ONOFF enable FIPS mode. In the future other values may specify additional actions beyond enabling FIPS mode, such as a value of 2 to designate an additional restriction to Suite B algorithms.
>
> Maybe this finally became true somehow. It is still puzzling why such an important change happened in a minor RHEL release and without a single word in their Release Notes.

I think the value 2 for Suite B is a very old thing, it has been documented in our man page since we added the *fips_mode* option with 5.0.4 (Suite B is deprecated, though, so no idea if there really is a difference now).

> I have a ticket opened with RedHat support, if they ever say anything, I'll update this ticket to keep it for posterity.

OK, great.

> This was a very helpful (and learning, albeit somewhat stressful) experience - thank you very much for your help! You may close this issue, I guess.

You're welcome. I'll close it once the fallback fix is merged.

#10 - 10.12.2019 10:33 - Tobias Brunner

- Status changed from Feedback to Closed

- Assignee set to Tobias Brunner

- Resolution set to Fixed