strongSwan - Bug #198

Charon dies at startup when using OpenSSL plugin under FIPS mode

29.06.2012 05:59 - Brian Pruss

Status:	Closed	**Start date:**	29.06.2012
Priority:	High	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	libstrongswan		
Target version:	5.0.1		
Affected version:	4.6.4	**Resolution:**	Fixed

Description

The following was found on an Red Hat Enterprise Linux 6.2 system, configured for FIPS mode per the instructions at http://goo.gl/9ifjD . The StrongSwan build is a slightly modified version of the EPEL 4.6.4 package from the testing line, rebuilt from the Source RPM and changed only to add the --enable-openssl plugin option to the autoconf options list.

If FIPS mode is enabled (fips=1 passed to the kernel command line, and /proc/sys/crypto/fips_enabled reads 1), charon is not starting. If FIPS mode is disabled (edit the kernel command line in grub to remove the fips=1 option), charon starts normally.

This appears to be related to the OpenSSL plugin. If the strongswan.conf file is modified to manually exclude the openssl plugin, charon starts normally.

Starter output is attached for three cases: FIPS/openssl-enabled, FIPS/openssl-disabled, Non-FIPS/openssl-enabled. The configuration is otherwise identical for each.

I believe that enabling FIPS mode in the kernel is causing the OPENSSL_config() call to implicitly enter FIPS mode (thus eliminating the need for a FIPS_mode_enter(1) call).

History

#1 - 29.06.2012 08:28 - Tobias Brunner

- Status changed from New to Feedback

Charon does not actually crash, but either hangs somewhere or just needs longer than usual to start, eventually starter kills it after 10 seconds.

Two things you can try:

1. Increase the time starter waits for charon in source:src/starter/invokecharon.c#L202 before it kills it.
2. Use *gdb* to find out if and where charon hangs:

```
# ipsec start --nofork
CTRL-Z to send starter into background so it does not kill charon
# gdb /usr/libexec/ipsec/charon `cat /var/run/charon.pid`
(gdb) thread apply all bt
```

Just adjust the paths if they are different for your build.

I believe that enabling FIPS mode in the kernel is causing the OPENSSL_config() call to implicitly enter FIPS mode.

Well, OPENSSL_config() does certainly call FIPS_mode_set() implicitly if FIPS mode is enabled in the OpenSSL config. You could try if disabling it explicitly makes any difference (if that's possible at all in case of the fips=1 kernel option).
From either of the two OpenSSL FIPS Module User Guides:

```
openssl_conf = XXXX_options
...
[ XXXX_options ]
alg_section = algs
...
[ algs ]
fips_mode = no
...
```

#2 - 30.06.2012 02:38 - Brian Pruss

Thanks for the quick response! I have more info on the problem:

First, the backtrace, as requested:

```
Thread 1 (Thread 0x7f34bd5db700 (LWP 18638)):
#0  0x00007f34bc8b0ff4 in __lll_lock_wait () from /lib64/libpthread.so.0
#1  0x00007f34bc8ac328 in _L_lock_854 () from /lib64/libpthread.so.0
#2  0x00007f34bc8ac1f7 in pthread_mutex_lock () from /lib64/libpthread.so.0
#3  0x00007f34bd1b6fca in lock (this=<value optimized out>)
    at threading/mutex.c:105
#4  0x00007f34ba5c34a5 in FIPS_mode () at fips.c:109
#5  0x00007f34ba5c393a in FIPS_mode_set (onoff=1) at fips.c:392
#6  0x00007f34ba4f0cda in init_fips_mode () at o_init.c:94
#7  OPENSSL_init_library () at o_init.c:112
#8  0x00007f34ba54c9c9 in ENGINE_load_builtin_engines () at eng_all.c:68
#9  0x00007f34ba5a74ba in OPENSSL_config (config_name=0x0) at conf_sap.c:85
#10 0x00007f34ba849838 in openssl_plugin_create () at openssl_plugin.c:412
#11 0x00007f34bd1b2af1 in create_plugin (this=<value optimized out>,
    handle=<value optimized out>, name=0x1ecd2a0 "openssl",
    integrity=<value optimized out>, entry=0x7ffff2f7d7e0)
    at plugins/plugin_loader.c:129
#12 0x00007f34bd1b3379 in load_plugin (this=0x1ec4590,
    path=0x7f34bd1c0816 "/usr/lib64/strongswan/plugins",
    list=<value optimized out>) at plugins/plugin_loader.c:179
#13 load_plugins (this=0x1ec4590,
    path=0x7f34bd1c0816 "/usr/lib64/strongswan/plugins",
    list=<value optimized out>) at plugins/plugin_loader.c:530
#14 0x00007f34bcd51184 in initialize (this=0x1ec4c20) at daemon.c:219
#15 0x0000000000401f26 in main ()
(gdb)
```

I dug into the problem some more, and I discovered the following. OpenSSL is calling the mutex lock function as seen in the backtrace, because it isn't seeing the charon thread as being the "fips owner" aka the top-level thread that did the FIPS entry call. It keeps a global "fips_thread" variable to keep track of this, and it uses a 0 value to indicate that it hasn't been set yet.

What I'm seeing, is that when Charon initializes the OpenSSL library, OpenSSL goes to initialize fips_thread and uses a callback to id_function() from openssl_plugin.c to get the value. However, id_function is returning 0 for some reason, and OpenSSL is happily writing that to fips_thread. Later, when it looks at that variable, it thinks that we're running under a subordinate thread, and tries to get a mutex that never gets put. Thus Charon stays blocked forever.

I'm not sure why the id is 0, given that I can see the code in libstrongswan/threading/thread.c where it's getting initialized to next_id, which is init to 1. Hints on where else to look would be welcome.

Here's the debug session where I saw the callback return 0:

```
[root@spgvm01 ~]# gdb --args /usr/libexec/strongswan/charon
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-50.el6)
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /usr/libexec/strongswan/charon...(no debugging symbols found)...done.
(gdb) break fips_set_owning_thread
Function "fips_set_owning_thread" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y

Breakpoint 1 (fips_set_owning_thread) pending.
(gdb) r
Starting program: /usr/libexec/strongswan/charon
[Thread debugging using libthread_db enabled]
00[DMN] Starting IKEv2 charon daemon (strongSwan 4.6.4)
00[CFG] loading ca certificates from '/etc/strongswan/ipsec.d/cacerts'
00[LIB] building CRED_CERTIFICATE - X509 failed, tried 2 builders
00[CFG]   loading ca certificate from '/etc/strongswan/ipsec.d/cacerts/AME_RSA_RootOSPGCUS.pem' failed
00[LIB] building CRED_CERTIFICATE - X509 failed, tried 2 builders
00[CFG]   loading ca certificate from '/etc/strongswan/ipsec.d/cacerts/AME_RSA_RootOSPGCUS.ppem' failed
00[CFG] loading aa certificates from '/etc/strongswan/ipsec.d/aacerts'
00[CFG] loading ocsp signer certificates from '/etc/strongswan/ipsec.d/ocspcerts'
00[CFG] loading attribute certificates from '/etc/strongswan/ipsec.d/acerts'
```

```
00[CFG] loading crls from '/etc/strongswan/ipsec.d/crls'
00[CFG] loading secrets from '/etc/strongswan/ipsec.secrets'
00[LIB] building CRED_PRIVATE_KEY - RSA failed, tried 2 builders
00[CFG]   loading private key from '/etc/strongswan/ipsec.d/private/swan_RSA_privkey.pem' failed

Breakpoint 1, fips_set_owning_thread () at fips.c:506
warning: Source file is more recent than executable.
506        {
Missing separate debuginfos, use: debuginfo-install strongswan-4.6.4-9MSI.el6.x86_64
(gdb) l
501            }
502        return ret;
503        }
504
505    int fips_set_owning_thread(void)
506        {
507        int ret = 0;
508
509        if (fips_is_started())
510            {
(gdb) n
509        if (fips_is_started())
(gdb)
511            CRYPTO_w_lock(CRYPTO_LOCK_FIPS2);
(gdb)
512            if (fips_thread == 0)
(gdb)
514                fips_thread = CRYPTO_thread_id();
(gdb) s
CRYPTO_thread_id () at cryptlib.c:534
534        {
(gdb)
537        if (id_callback == NULL)
(gdb)
554        }
(gdb)
552            ret=id_callback();
(gdb)
id_function () at openssl_plugin.c:140
140    {
(gdb)
141        return (unsigned long)thread_current_id();
(gdb) p thread_current_id
$1 = {u_int ()} 0x7ffff7bc34e0 <thread_current_id>
(gdb) s
thread_current_id () at threading/thread.c:347
347    {
(gdb)
348        private_thread_t *this = (private_thread_t*)thread_current();
(gdb) p current_thread
$2 = (thread_value_t *) 0x604280
(gdb) p current_thread->get(current_thread)
$3 = (void *) 0x6040a0
(gdb) s
thread_current () at threading/thread.c:340
340        return current_thread->get(current_thread);
(gdb) s
get (this=0x604280) at threading/thread_value.c:47
47        return pthread_getspecific(this->key);
(gdb) s
0x00007ffff72bc420 in pthread_getspecific () from /lib64/libpthread.so.0
(gdb) s
Single stepping until exit from function pthread_getspecific,
which has no line number information.
thread_current_id () at threading/thread.c:350
350        return this ? this->id : 0;
(gdb) p this
$4 = (private_thread_t *) 0x6040a0
(gdb) p this->id
$5 = 0
(gdb)
```

Thanks again for your help so far.

#3 - 30.06.2012 10:13 - Tobias Brunner

- File 0001-openssl-Ensure-the-thread-ID-is-never-zero.patch added

Thanks for the details.

> However, id_function is returning 0 for some reason, and OpenSSL is happily writing that to fips_thread.

This is because thread_current_id() does not use pthread_self() but instead returns a custom thread id that starts at 0 for the main thread (which is the thread initializing the openssl plugin). This is more user-friendly when reading logs, but here it is obviously not such a good idea. But this is only part of the problem...

> Later, when it looks at that variable, it thinks that we're running under a subordinate thread, and tries to get a mutex that never gets put.

The reason this happens is because OpenSSL locks the mutex in FIPS_mode_set() and then tries to lock it again in FIPS_mode() - only because of the unfortunate thread_id above - which does then not work because the mutexes are not created with recursive locking enabled (i.e. they block even if the thread currently holding it is the same).

Could you please try if the attached patch fixes the problem.

#4 - 01.07.2012 00:57 - Brian Pruss

I've applied the patch, and I'm happy to report that it seems to resolve the problem. Any chance this can make it into the next release?

Thanks again for the quick turnaround on this.

#5 - 03.07.2012 12:03 - Tobias Brunner

- Category set to libstrongswan

- Status changed from Feedback to Closed

- Assignee set to Tobias Brunner

- Target version set to 5.0.1

- Resolution set to Fixed

Thanks for testing.

Unfortunately, 5.0.0 was just released last Saturday so this fix has to wait until the next release.

Files

charon_start_failure_openssl_fips_mode.txt	2.25 KB	29.06.2012	Brian Pruss
charon_start_success_fips_mode_disabled_openssl_plugin_enabled.txt	2.67 KB	29.06.2012	Brian Pruss
charon_start_success_openssl_plugin_disabled.txt	2.62 KB	29.06.2012	Brian Pruss
0001-openssl-Ensure-the-thread-ID-is-never-zero.patch	1.1 KB	30.06.2012	Tobias Brunner