strongSwan - Issue #2750

setting WFP SA SPI failed: 0x80320035

07.09.2018 16:38 - Mickael Chazaux

Status:	Feedback		
Priority:	Normal		
Assignee:			
Category:			
Affected version:	5.6.3	**Resolution:**	

Description

Hello,

I am testing strongswan on two Windows server 2012 r2 boxes.

Sometimes adding a child SA works, sometimes it fails. Specifically, the IPsecSaContextSetSpi0 function fails with code FWP_E_INVALID_PARAMETER.

the three chunks of traces below show the cases :
- Works on both sides,
- Fails on the remote side,
- Fails on the local side.

Any ideas?

```
Administrateur@s1 /cygdrive/c/strongswan
$ ./swanctl.exe -i -c trap-yn
[IKE] establishing CHILD_SA trap-yn{3}
[ENC] generating CREATE_CHILD_SA request 4 [ N(USE_TRANSP) SA No TSi TSr ]
[NET] sending packet: from 192.168.38.14[4500] to 192.168.38.47[4500] (272 bytes)
[NET] received packet: from 192.168.38.47[4500] to 192.168.38.14[4500] (208 bytes)
[ENC] parsed CREATE_CHILD_SA response 4 [ N(USE_TRANSP) SA No TSi TSr ]
[IKE] CHILD_SA trap-yn{3} established with SPIs c77e637e_i cc6d7174_o and TS 192.168.38.14/32 === 192.168.38.47/32
initiate completed successfully

Administrateur@s1 /cygdrive/c/strongswan
$ ./swanctl.exe -i -c trap-yn
[IKE] establishing CHILD_SA trap-yn{4}
[ENC] generating CREATE_CHILD_SA request 5 [ N(USE_TRANSP) SA No TSi TSr ]
[NET] sending packet: from 192.168.38.14[4500] to 192.168.38.47[4500] (272 bytes)
[NET] received packet: from 192.168.38.47[4500] to 192.168.38.14[4500] (80 bytes)
[ENC] parsed CREATE_CHILD_SA response 5 [ N(TS_UNACCEPT) ]
[IKE] received TS_UNACCEPTABLE notify, no CHILD_SA built
[IKE] failed to establish CHILD_SA, keeping IKE_SA
initiate failed: establishing CHILD_SA 'trap-yn' failed

Administrateur@s1 /cygdrive/c/strongswan
$ ./swanctl.exe -i -c trap-yn
[IKE] establishing CHILD_SA trap-yn{5}
[ENC] generating CREATE_CHILD_SA request 6 [ N(USE_TRANSP) SA No TSi TSr ]
[NET] sending packet: from 192.168.38.14[4500] to 192.168.38.47[4500] (272 bytes)
[NET] received packet: from 192.168.38.47[4500] to 192.168.38.14[4500] (208 bytes)
[ENC] parsed CREATE_CHILD_SA response 6 [ N(USE_TRANSP) SA No TSi TSr ]
[KNL] setting WFP SA SPI failed: 0x80320035
[IKE] unable to install IPsec policies (SPD) in kernel
[IKE] failed to establish CHILD_SA, keeping IKE_SA
[IKE] sending DELETE for ESP CHILD_SA with SPI c92acf11
[ENC] generating INFORMATIONAL request 7 [ D ]
[NET] sending packet: from 192.168.38.14[4500] to 192.168.38.47[4500] (80 bytes)
[NET] received packet: from 192.168.38.47[4500] to 192.168.38.14[4500] (80 bytes)
[ENC] parsed INFORMATIONAL response 7 [ D ]
initiate failed: establishing CHILD_SA 'trap-yn' failed
```

History

#1 - 10.09.2018 11:05 - Tobias Brunner

- Status changed from New to Feedback

> Any ideas?

Not really. The handle and sa_id should be valid (the former was used to successfully create the latter in the previous call), the addresses set are probably always the same in your test, so that leaves the SPI. Did you try logging the SPIs that are set there?

#2 - 10.09.2018 11:54 - Mickael Chazaux

Hello,

I've been searching and found some links saying that certain SPI values are refused by the Windows API.

Following a hunch, I've been testing all morning forcing certain bits from the SPI to 0 or 1. With this (crappy) patch it works all the time :

```
--- ../strongswan/src/libcharon/plugins/kernel_wfp/kernel_wfp_ipsec.c    2018-09-07 16:46:02.721192458 +0200
+++ src/libcharon/plugins/kernel_wfp/kernel_wfp_ipsec.c    2018-09-10 11:44:45.964042830 +0200
@@ -1991,7 +1991,13 @@
      * in, and it must satisfy p % 4 == 3 to map x > p/2 using p - qr. */
     static const u_int p = 268435399, offset = 0xc0000000;

-    *spi = htonl(offset + permute(ref_get(&this->nextspi) ^ this->mixspi, p));
+    uint32_t s = offset + permute(ref_get(&this->nextspi) ^ this->mixspi, p);
+
+    s &= ~(1 << 0);
+
+    printf("Setting SPI bit 0 to 0\n", p2);
+
+    *spi = htonl(s);
     return SUCCESS;
 }
```

Worked 25 times in a row. Without this, it works only around one in four tries.
My explanation is 1/4 is (1/2 chance to get a even SPI locally) times (1/2 chance to get a even SPI remotely).

The SPI must be even to be accepted !!??

FYI, I am using Transport mode. Maybe Tunnel mode has the Odd values ?

Best regards,

Mickael

#3 - 10.09.2018 14:27 - Tobias Brunner

> The SPI must be even to be accepted !!??

Sounds weird. Did you log the SPIs that are generated by the existing code? To see if the failure only occurs if the SPI is odd?

> FYI, I am using Transport mode. Maybe Tunnel mode has the Odd values ?

Try it out :)

#4 - 10.09.2018 15:57 - Mickael Chazaux

I did the logging thing :

```
--- ../strongswan/src/libcharon/plugins/kernel_wfp/kernel_wfp_ipsec.c    2018-09-07 16:46:02.721192458 +0200
+++ src/libcharon/plugins/kernel_wfp/kernel_wfp_ipsec.c    2018-09-10 15:26:46.624254025 +0200
@@ -1202,11 +1202,13 @@
                                 ntohl(entry->isa.spi));
     if (res != ERROR_SUCCESS)
     {
```

```
+        DBG1(DBG_KNL, "IPsecSaContextSetSpi0 Failed, spi %08x (%s)\n", ntohl(entry->isa.spi), (ntohl(entry->i
sa.spi) & 1) ? "odd":"even");
         DBG1(DBG_KNL, "setting WFP SA SPI failed: 0x%08x", res);
         IPsecSaContextDeleteById0(this->handle, entry->sa_id);
         entry->sa_id = 0;
         return FALSE;
     }
+    DBG1(DBG_KNL, "IPsecSaContextSetSpi0 Succeeded, spi %08x (%s)\n", ntohl(entry->isa.spi), (ntohl(entry->is
a.spi) & 1) ? "odd":"even");

     if (!install_sa(this, entry, TRUE, &entry->isa, spi.ipVersion) ||
         !install_sa(this, entry, FALSE, &entry->osa, spi.ipVersion))
```

Then grepping :

```
$ grep 'Failed\|Succeeded' log.txt
11[KNL] IPsecSaContextSetSpi0 Succeeded, spi cb968ed2 (even)
11[KNL] IPsecSaContextSetSpi0 Failed, spi c696e997 (odd)
14[KNL] IPsecSaContextSetSpi0 Succeeded, spi c84175fe (even)
13[KNL] IPsecSaContextSetSpi0 Failed, spi cb3efcb1 (odd)
05[KNL] IPsecSaContextSetSpi0 Succeeded, spi cce98944 (even)
15[KNL] IPsecSaContextSetSpi0 Failed, spi c7e9e391 (odd)
11[KNL] IPsecSaContextSetSpi0 Succeeded, spi c9947020 (even)
16[KNL] IPsecSaContextSetSpi0 Succeeded, spi c1e92f42 (even)
06[KNL] IPsecSaContextSetSpi0 Failed, spi c393bbdd (odd)
13[KNL] IPsecSaContextSetSpi0 Failed, spi ce9415d9 (odd)
12[KNL] IPsecSaContextSetSpi0 Failed, spi c03ea2a9 (odd)
10[KNL] IPsecSaContextSetSpi0 Succeeded, spi cdea9828 (even)
06[KNL] IPsecSaContextSetSpi0 Failed, spi cf9524ab (odd)
13[KNL] IPsecSaContextSetSpi0 Succeeded, spi ca957f28 (even)
16[KNL] IPsecSaContextSetSpi0 Failed, spi cc400ba7 (odd)
08[KNL] IPsecSaContextSetSpi0 Failed, spi c494ca79 (odd)
14[KNL] IPsecSaContextSetSpi0 Succeeded, spi c63f5704 (even)
05[KNL] IPsecSaContextSetSpi0 Failed, spi c13fb169 (odd)
16[KNL] IPsecSaContextSetSpi0 Succeeded, spi c2ea3df0 (even)
09[KNL] IPsecSaContextSetSpi0 Succeeded, spi cb44a934 (even)
13[KNL] IPsecSaContextSetSpi0 Failed, spi ccef3567 (odd)
07[KNL] IPsecSaContextSetSpi0 Succeeded, spi c7ef90d4 (even)
16[KNL] IPsecSaContextSetSpi0 Failed, spi c99a1d03 (odd)
11[KNL] IPsecSaContextSetSpi0 Failed, spi c1eeda45 (odd)
13[KNL] IPsecSaContextSetSpi0 Succeeded, spi c3996680 (even)
12[KNL] IPsecSaContextSetSpi0 Succeeded, spi ce99c19c (even)
09[KNL] IPsecSaContextSetSpi0 Succeeded, spi c0444e0c (even)
10[KNL] IPsecSaContextSetSpi0 Failed, spi cdf047ab (odd)
14[KNL] IPsecSaContextSetSpi0 Succeeded, spi cf9ad3ce (even)
05[KNL] IPsecSaContextSetSpi0 Failed, spi ca9b2f6b (odd)
06[KNL] IPsecSaContextSetSpi0 Succeeded, spi cc45bb8a (even)
13[KNL] IPsecSaContextSetSpi0 Succeeded, spi c49a787c (even)
07[KNL] IPsecSaContextSetSpi0 Failed, spi c64504a7 (odd)
16[KNL] IPsecSaContextSetSpi0 Succeeded, spi c145602c (even)
12[KNL] IPsecSaContextSetSpi0 Failed, spi c2efec53 (odd)
08[KNL] IPsecSaContextSetSpi0 Succeeded, spi c5ed6dc6 (even)
15[KNL] IPsecSaContextSetSpi0 Failed, spi c797fa19 (odd)
10[KNL] IPsecSaContextSetSpi0 Succeeded, spi c2985526 (even)
13[KNL] IPsecSaContextSetSpi0 Failed, spi c442e175 (odd)
16[KNL] IPsecSaContextSetSpi0 Succeeded, spi cc979f1e (even)
10[KNL] IPsecSaContextSetSpi0 Failed, spi ce422b79 (odd)
09[KNL] IPsecSaContextSetSpi0 Succeeded, spi c942866e (even)
11[KNL] IPsecSaContextSetSpi0 Failed, spi caed12c5 (odd)
05[KNL] IPsecSaContextSetSpi0 Failed, spi c8990b3d (odd)
10[KNL] IPsecSaContextSetSpi0 Succeeded, spi ca439780 (even)
13[KNL] IPsecSaContextSetSpi0 Failed, spi c543f2bd (odd)
11[KNL] IPsecSaContextSetSpi0 Succeeded, spi c6ee7efc (even)
09[KNL] IPsecSaContextSetSpi0 Failed, spi cf433c55 (odd)
12[KNL] IPsecSaContextSetSpi0 Failed, spi c0edc8d9 (odd)
13[KNL] IPsecSaContextSetSpi0 Failed, spi cbee23c5 (odd)
06[KNL] IPsecSaContextSetSpi0 Succeeded, spi cd98b00c (even)
05[KNL] IPsecSaContextSetSpi0 Succeeded, spi cb39622e (even)
10[KNL] IPsecSaContextSetSpi0 Failed, spi cce3ef21 (odd)
11[KNL] IPsecSaContextSetSpi0 Succeeded, spi c7e4484e (even)
16[KNL] IPsecSaContextSetSpi0 Failed, spi c98ed53d (odd)
06[KNL] IPsecSaContextSetSpi0 Failed, spi c1e3963f (odd)
07[KNL] IPsecSaContextSetSpi0 Succeeded, spi c38e233a (even)
09[KNL] IPsecSaContextSetSpi0 Succeeded, spi ce8e7c16 (even)
06[KNL] IPsecSaContextSetSpi0 Succeeded, spi c0390946 (even)
```

```
07[KNL] IPsecSaContextSetSpi0 Failed, spi cde4faa5 (odd)
16[KNL] IPsecSaContextSetSpi0 Succeeded, spi cf8f8788 (even)
12[KNL] IPsecSaContextSetSpi0 Failed, spi ca8fe0e5 (odd)
07[KNL] IPsecSaContextSetSpi0 Succeeded, spi cc3a6dc4 (even)
07[KNL] IPsecSaContextSetSpi0 Succeeded, spi c48f2e76 (even)
14[KNL] IPsecSaContextSetSpi0 Failed, spi c639bb61 (odd)
08[KNL] IPsecSaContextSetSpi0 Succeeded, spi c13a14a6 (even)
10[KNL] IPsecSaContextSetSpi0 Failed, spi c2e4a18d (odd)
13[KNL] IPsecSaContextSetSpi0 Succeeded, spi c5e232c0 (even)
05[KNL] IPsecSaContextSetSpi0 Failed, spi c78cbfd3 (odd)
09[KNL] IPsecSaContextSetSpi0 Succeeded, spi c28d18a0 (even)
11[KNL] IPsecSaContextSetSpi0 Failed, spi c437a5af (odd)
14[KNL] IPsecSaContextSetSpi0 Succeeded, spi cc8c6718 (even)
05[KNL] IPsecSaContextSetSpi0 Failed, spi ce36f433 (odd)
12[KNL] IPsecSaContextSetSpi0 Succeeded, spi c9374ce8 (even)
07[KNL] IPsecSaContextSetSpi0 Failed, spi cae1d9ff (odd)
06[KNL] IPsecSaContextSetSpi0 Failed, spi c88dca37 (odd)
15[KNL] IPsecSaContextSetSpi0 Succeeded, spi ca38573a (even)
06[KNL] IPsecSaContextSetSpi0 Failed, spi c538b037 (odd)
16[KNL] IPsecSaContextSetSpi0 Succeeded, spi c6e33d36 (even)
13[KNL] IPsecSaContextSetSpi0 Failed, spi cf37fe4f (odd)
07[KNL] IPsecSaContextSetSpi0 Failed, spi c0e28b93 (odd)
08[KNL] IPsecSaContextSetSpi0 Failed, spi cbe2e43f (odd)
09[KNL] IPsecSaContextSetSpi0 Succeeded, spi cd8d7146 (even)
12[KNL] IPsecSaContextSetSpi0 Failed, spi c5e7c743 (odd)
08[KNL] IPsecSaContextSetSpi0 Succeeded, spi c79253f6 (even)
16[KNL] IPsecSaContextSetSpi0 Failed, spi c292ade3 (odd)
10[KNL] IPsecSaContextSetSpi0 Succeeded, spi c43d3a92 (even)
07[KNL] IPsecSaContextSetSpi0 Failed, spi cc91fa1b (odd)
13[KNL] IPsecSaContextSetSpi0 Succeeded, spi ce3c86d6 (even)
07[KNL] IPsecSaContextSetSpi0 Failed, spi c93ce0ab (odd)
11[KNL] IPsecSaContextSetSpi0 Succeeded, spi cae76d62 (even)
15[KNL] IPsecSaContextSetSpi0 Succeeded, spi c89361ba (even)
08[KNL] IPsecSaContextSetSpi0 Failed, spi ca3dee5d (odd)
15[KNL] IPsecSaContextSetSpi0 Succeeded, spi c53e487a (even)
08[KNL] IPsecSaContextSetSpi0 Failed, spi c6e8d519 (odd)
14[KNL] IPsecSaContextSetSpi0 Succeeded, spi cf3d9452 (even)
10[KNL] IPsecSaContextSetSpi0 Succeeded, spi c0e82136 (even)
10[KNL] IPsecSaContextSetSpi0 Succeeded, spi cbe87b02 (even)
```

Well...

About the Transport mode. I do not think the point is to reverse engineer what values are acceptable to Windows as SPI and what are not. The examples on MSDN do not use the IPsecSaContextSetSpi0 function but rather the IPsecSaContextGetSpi0.

Is it possible to call the IPsecSaContextGetSpi0 from get_spi() to get a windows-valid SPI, then IPsecSaContextSetSpi0 from install_sa() to set the actual SA parameters?

I can't figure out from where the get_spi() func is called. I suppose it is called before exchanging the SPIs with the peer, and before install_sa().

I see two solutions to this problem:

Solution 1 :

```
- Find out what SPI values are refused
 - Filter the output of the current get_spi to only generate valid SPIs
```

Of course the best would be a document stating what the function does. Undefined behaviour bad.
Maybe a disassembly of IPsecSaContextSetSpi0 would help to see on what grounds it decides to throw an error.
Or simply like I did, brute-force the function and find a pattern.

Solution 2 :

```
- call IPsecSaContextGetSpi0 in get_spi
 - Do SPI exchange
 - call IPsecSaContextSetSpi0 in install_sa()
```

But the Wiki says :

```
https://wiki.strongswan.org/projects/strongswan/wiki/kernel-wfp

State and SPI management

The strongSwan IPsec kernel abstraction layer is loosely defined for the PF_KEY and similar interfaces, workin
g on SA and policy granular level. It usually installs in- and outbound Security Associations, and then attach
```

```
es IPsec policies using a unique reqid identifier.

For the WFP interface, this paradigm does not work very well. It expects the policy information first, followe
d by the SA context and the in- and outbound Security Associations. The WFP interface can't allocate SPIs with
out creating policy entries and an SA context first. In strongSwan, however, we must allocate SPIs before inst
alling policies, as this information is negotiated in IKEv2.

To work around these interface differences, the kernel-wfp plugin caches any policy or SA addition. Once all t
his information has been collected, it installs it in a single batch. Instead of using kernel allocated SPIs u
sing IPsecSaContextGetSpi0, it allocates SPIs in userland pseudo-randomly and uses IPsecSaContextSetSpi0 to se
t it.

All policy and SA information installed by strongSwan is non-persistent. A system reboot or a restart of the B
ase Filtering Engine service removes any rule installed by the plugin.
```

Please comment.

Best regards,

Mickael

#5 - 10.09.2018 16:56 - Tobias Brunner

> I did the logging thing :
>
> [...]
>
> Then grepping :
>
> [...]
>
> Well...

Interesting. It still seems totally weird (what could their rationale have been for this?). Did you try other (i.e. newer) Windows versions?

> Is it possible to call the IPsecSaContextGetSpi0 from get_spi() to get a windows-valid SPI, then IPsecSaContextSetSpi0 from install_sa() to set the actual SA parameters?

No, as you found via our wiki, it's not (at least that's my understanding).

> I can't figure out from where the get_spi() func is called. I suppose it is called before exchanging the SPIs with the peer, and before install_sa().

Yes, it's called before the IKE_AUTH/CREATE_CHILD_SA exchange is sent.

> I see two solutions to this problem:
>
> Solution 1 :
>
> - Find out what SPI values are refused
> - Filter the output of the current get_spi to only generate valid SPIs
>
> Of course the best would be a document stating what the function does. Undefined behaviour bad.
> Maybe a disassembly of IPsecSaContextSetSpi0 would help to see on what grounds it decides to throw an error.
> Or simply like I did, brute-force the function and find a pattern.

As usual, the documentation is quite lacking (no mention of any special restrictions on the SPIs). How about contacting Microsoft about it?

> Solution 2 :
>
> - call IPsecSaContextGetSpi0 in get_spi
> - Do SPI exchange
> - call IPsecSaContextSetSpi0 in install_sa()

You already found that this probably doesn't work. (I guess it might be possible to delete the context again and use the first call just to get a valid SPI value, but that doesn't seem like an improvement).

#6 - 10.09.2018 17:22 - Mickael Chazaux

Interesting things popup wih gdb :

```
$ gdb FWPUCLNT.DLL
[blah]
(gdb) disassemble IPsecSaContextSetSpi0
Dump of assembler code for function fwpuclnt!IPsecSaContextSetSpi0:
   0x0000000180032080 <+0>:     mov    %r9d,0x20(%rsp)
   0x0000000180032085 <+5>:     push   %rbx
   0x0000000180032086 <+6>:     sub    $0x30,%rsp
   0x000000018003208a <+10>:    xor    %ebx,%ebx
   0x000000018003208c <+12>:    test   %r9d,%r9d
   0x000000018003208f <+15>:    je     0x1800320ea <fwpuclnt!IPsecSaContextSetSpi0+106>
   0x0000000180032091 <+17>:    test   $0x1,%r9b
   0x0000000180032095 <+21>:    jne    0x1800320ea <fwpuclnt!IPsecSaContextSetSpi0+106>
[...]
   0x00000001800320ea <+106>:   mov    $0x80320035,%eax
   0x00000001800320ef <+111>:   add    $0x30,%rsp
   0x00000001800320f3 <+115>:   pop    %rbx
   0x00000001800320f4 <+116>:   retq
```

I am no assembly expert, but the "test $0x1,%r9b" looks like what I am observing. I interpret this as "And 1 and r9b, jump to offset 106 if non zero, then load error code 0x80320035 in eax, then return."

This is the dll I took from a Windows Server 2012-r2.

It looks the same on W10 :

```
Dump of assembler code for function fwpuclnt!IPsecSaContextSetSpi0:
   0x100218b0 <+0>:     mov    %edi,%edi
   0x100218b2 <+2>:     push   %ebp
   0x100218b3 <+3>:     mov    %esp,%ebp
   0x100218b5 <+5>:     cmpl   $0x0,0x18(%ebp)
   0x100218b9 <+9>:     je     0x10021919 <fwpuclnt!IPsecSaContextSetSpi0+105>
   0x100218bb <+11>:    testb  $0x1,0x18(%ebp)
   0x100218bf <+15>:    jne    0x10021919 <fwpuclnt!IPsecSaContextSetSpi0+105>
[..]
   0x10021919 <+105>:   mov    $0x80320035,%eax
   0x1002191e <+110>:   pop    %ebp
   0x1002191f <+111>:   ret    $0x14
```

#7 - 10.09.2018 18:21 - Mickael Chazaux

Hello,

I do not know what to do now. Should I submit a patch for the get_spi function ? Is it likely to be accepted in strongSwan ?

I can try to write to Microsoft via our MSDN account.

Best regards,

Mickael

#8 - 11.09.2018 12:41 - Tobias Brunner

> Should I submit a patch for the get_spi function ? Is it likely to be accepted in strongSwan ?

I suppose. But while there does seem to be such a check in the Windows code, this really doesn't make much sense (the only reserved SPIs are those between 0 and 255, according to RFC 4301). And I wonder why you're the first to report this issue (the customer for whom we did the Windows port all those years ago had/has this in widespread use, and Martin, apparently, also didn't stumble across it during development). Did you find any other reports, besides this unanswered question from last year, and this one claiming the SPIs have to be >> 2^16 (which they are anyway because they start at 0xc0000000)?

> I can try to write to Microsoft via our MSDN account.

That'd be great. How hopeful should we be that we get an answer or even a fix? Do you have any experience?

#9 - 27.05.2019 11:37 - Valtteri Vuorikoski

I can confirm this issue on Windows 10 Pro build 17763 and strongSwan 4.8.0. Mickael's patch fixes error 0x80320035, but ends up with another error (in new ticket). It occurs when trying to establish IPv6 transport mode SAs. I didn't try any other SA types.

I let IKEEXT create a bunch of SAs and they had all LSB set to 0, so I think this patch should be adopted.

#10 - 27.05.2019 11:59 - Tobias Brunner

> I think this patch should be adopted.

It still makes absolutely no sense and I haven't seen any reasonable explanation for it. But I guess we could theoretically add support for *charon.spi_min/max* (supported for other kernel interfaces since 5.5.2), which would allow avoiding some SPIs.