ExaTrack

Chez moi, ça marche.

Beerump 2022

Clément "hakril" Rouault 16/09/2022

C’est l’histoire d’un bug

- Dans PythonForWindows (Issue 10)

create_thread & NtCreateThreadEx_32_to_64 #10

Closed quentinhardy opened this issue on Jan 16, 2020 · 10 comments

quentinhardy commented on Jan 16, 2020

Hello,

I am using python 2.7 (X86 version) and the version 0.5 of PythonForWindows (on pip) on Windows 10.

When I try to use *create_thread()* from **x86** process, more exactly ***NtCreateThreadEx_32_to_64()***, for creating/executing a new thread on (another) **x64** process, I have a the following error:

```
[...]
  File ".\test.py", line 193, in testRemoteThread
    winProcess.create_thread(addr=reflectiveLoader, param=0)
  File "C:\Python27\lib\site-packages\pythonforwindows-0.5-py2.7.egg\windows\winobject\process.py", line 1040, in create_
    windows.syswow64.NtCreateThreadEx_32_to_64(ThreadHandle=byref(thread_handle) ,ProcessHandle=self.handle, lpStartAddre
  File "C:\Python27\lib\site-packages\pythonforwindows-0.5-py2.7.egg\windows\syswow64.py", line 260, in NtCreateThreadEx_
    return NtCreateThreadEx_32_to_64.ctypes_function(ThreadHandle, DesiredAccess, ObjectAttributes, ProcessHandle, lpStar
  File "C:\Python27\lib\site-packages\pythonforwindows-0.5-py2.7.egg\windows\syswow64.py", line 232, in perform_call
    return self.raw_call(*args)
  File "C:\Python27\lib\site-packages\pythonforwindows-0.5-py2.7.egg\windows\syswow64.py", line 138, in wrapper
    return native_caller()
WindowsError: exception: access violation writing 0x00000000
```

The targeted process is a *notepad.exe* (x64) for example.
There is no one error before this function when I use *virtual_alloc()*, *write_memory()* and *virtual_protect()* for example.

Do you have an idea? There is a problem with *NtCreateThreadEx_32_to_64()* from x86 process to x64 process?

Thank you in advance for your help,

Regards,

C’est l’histoire d’un bug

- Quand un process 32 (python) veut créer un thread dans un processus 64

```
process.create_thread(targetaddr)
windows.syswow64.NtCreateThreadEx_32_to_64([…])
[de la boue]
return native_caller()
```

```
WindowsError: exception: access violation
writing 0x00000000
```

Heaven’s gate en 1 slide

- Processus 32bits dans un windows 64 bit
  - Wow64 (Windows on Windows64)
  - Couche d’émulation
  - Presence de ntdll32 ET ntdll64 dans le processus
  - Possibilité de changer de bitness d’exécution en changeant le segment selector CS

Un Processus 32bits peut donc bypass tout ça en changeant son CS et en appelant des fonctions de ntdll64 manuellement !

Heaven’s gate en 2 slides

```
0:  [x86] 9a 0a 00 00 00 33 00    call   0x33:0xa
7:  [x86] c3                      ret
8:  [x86] 90                      nop
9:  [x86] 90                      nop
a:  [x64] 48 c7 c0 41 00 00 00    mov    rax,0x41
11: [x64] 48 ff c0                inc    rax
14: [x64] cb                      retf
```

`windows.syswow64.NtCreateThreadEx_32_to_64`

- Va appeler `<ntdll64!NtCreateThreadEx>` avec cette méthode !

Revenons à notre bug

- Je n'arrive pas à reproduire
  - « Chez moi, ça marche »
- Plein de théories, testées à coup de scripts exécutés chez quelqu'un que je ne connais pas
  - Un paramètre mal passé entre 32 / 64 bits ?
  - Un bug dans mon assembleur qui JIT les code 32 et 64 ?
- Tout semble pété chez lui
  - « Pourtant, chez moi, ça marche !»
  - « Je sais coder quand même ? »

Et chez les autres ?

- Envoie des scripts de tests à plusieurs potes
  - Heurs : « Chez moi, ça marche »
  - W4kfu : « Chez moi, ça marche »
  - Bruno : « Chez moi, ça marche »

Et bien voilà ! Je sais coder quand même !

Et chez les clients ?

- À ce moment, grosse mission
  - Mission de 1200 machines en base
  - Notre agent de collecte :
    - En Python
    - Collecte les exceptions Python (« ça pourra être utile un jour »)
    - Les exceptions sont dans la DB

Et chez les clients ?

```
# Retrieve access violation writing Exceptions
>>> ERROR_MESSAGE = "access violation writing 0x00000000"
>>> ee = query(AgentException).filter(AgentException.exc.contains(ERROR_MESSAGE))
>>> len(ee)
150341
# Le bug est apparu plus de 150 mille fois sur les collectes
```

Et chez les clients ?

```
# Retrieve the computer that triggered the bug
>>> compname = set(e.computer.name for e in ee)
>>> len(compname)
401
# Le bug est apparu sur 400 machines sur les 1200 collectées
```

On avance !

J’ai des machines ou ça ne marche pas !

Mauvaise foi et révélation

```
# le CPU de la première machine qui a eu le bug
>>> ee[0].computer.wmi["Win32_Processor"][0].name
'AMD PRO A10-8730B R5, 10 COMPUTE CORES 4C+6G   '

# le CPU de toutes les machines qui ont eu le bug
>>> BUG_PROC = set(x.name for x in
query(WMI_Win32_Processor).join(Computer).filter(Computer.name.in_(compname)))
# le CPU de toutes les machines qui n'ont PAS EU le bug
>>> NOBUG_PROC = set(x.name for x in
query(WMI_Win32_Processor).join(Computer).filter(Computer.name.notin_(compname
)))
```

Mauvaise foi et révélation

```
>>> BUG_PROC
{'AMD PRO A12-8800B R7, 12 Compute Cores 4C+8G', 'AMD
PRO A10-8730B R5, 10 COMPUTE CORES 4C+6G', 'AMD PRO
A8-8600B R6, B R7, 10 Compute Cores 4C+6G    '}
>>> NOBUG_PROC
{'Intel(R) Xeon(R) CPU E5-2403 v2 @ 1.80GHz', 'Intel(R)
Core(TM) i5-4200U CPU @ 1.60GHz', 'Intel(R) Xeon(R) CPU
X5650  @ 2.67GHz', 'Intel(R) Xeon(R) Gold 6128 CPU @
3.40GHz', 'Intel(R) Xeon(R) CPU E31240 @ 3.30GHz', 'Intel(R)
Xeon(R) CPU E5-2650 v2 @ 2.60GHz', 'Intel(R) Xeon(R) CPU E3-
1220 v5 @ 3.00GHz', 'Intel(R) Core(TM) i5-8250U CPU @
1.60GHz', 'Intel(R) Xeon(R) CPU           3040  @ 1.86GHz', … }
```

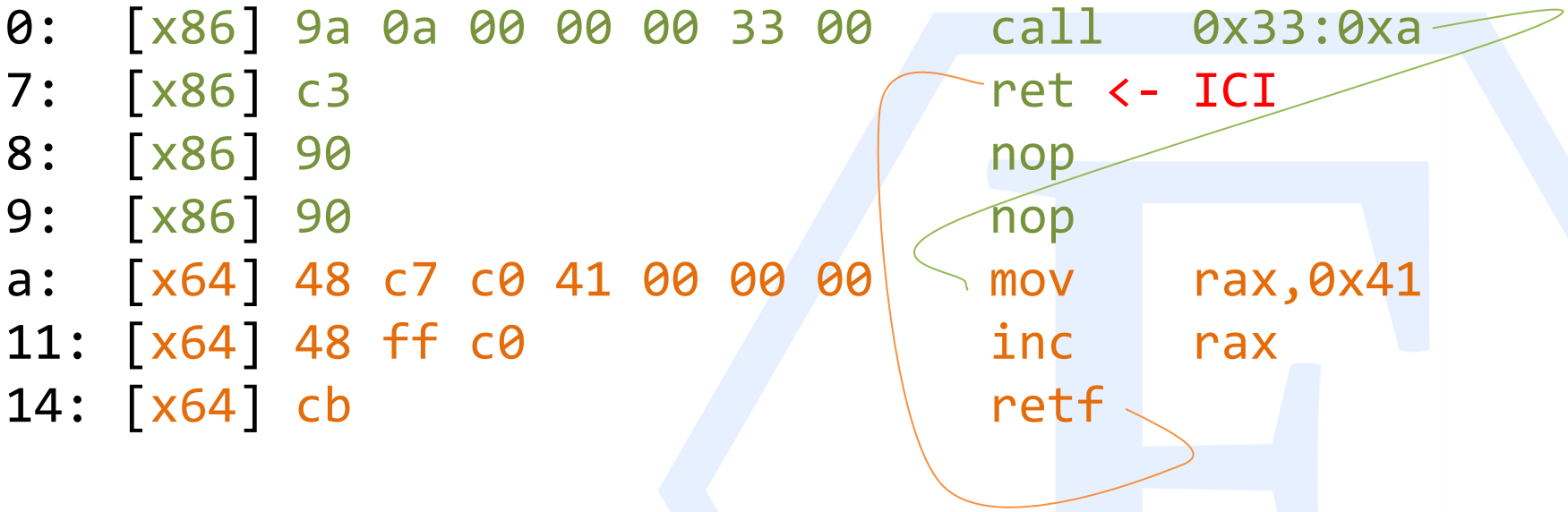
Fnac-Debugging
ExAMD

CARACTÉRISTIQUES

Marque du processeur	AMD
Modèle du processeur	AMD A4
Processeur	AMD A4-9120e
Mémoire cache du processeur	1 Mo
Fréquence du processeur	De 1.5 à 2.2 GHz
Caractéristiques de l'écran	HD SVA BrightView, rétroéclairage WLED
Capacité de stockage	64 Go

Identification

- Le bug trigger sur ExAMD !
  - Segfault sur le return APRÈS le retour en 32bits



Identification++

- Segfault sur la première opération de stack au retour 32bits
  - ESP ?

C’est un bug CPU

Tentative de fix en guess

```
0:  [x86] 9a 0c 00 00 00 33 00    call   0x33:0xc
7:  [x86] 8c d1             FIX-> mov    ecx,ss
9:  [x86] 8e d1             FIX-> mov    ss,ecx
b:  [x86] c3                      ret
a:  [x64] 48 c7 c0 41 00 00 00    mov    rax,0x41
11: [x64] 48 ff c0                inc    rax
14: [x64] cb                      retf
```

- Ça fonctionne !
  - probablement un problème de ss interne au passage 64 -> 32

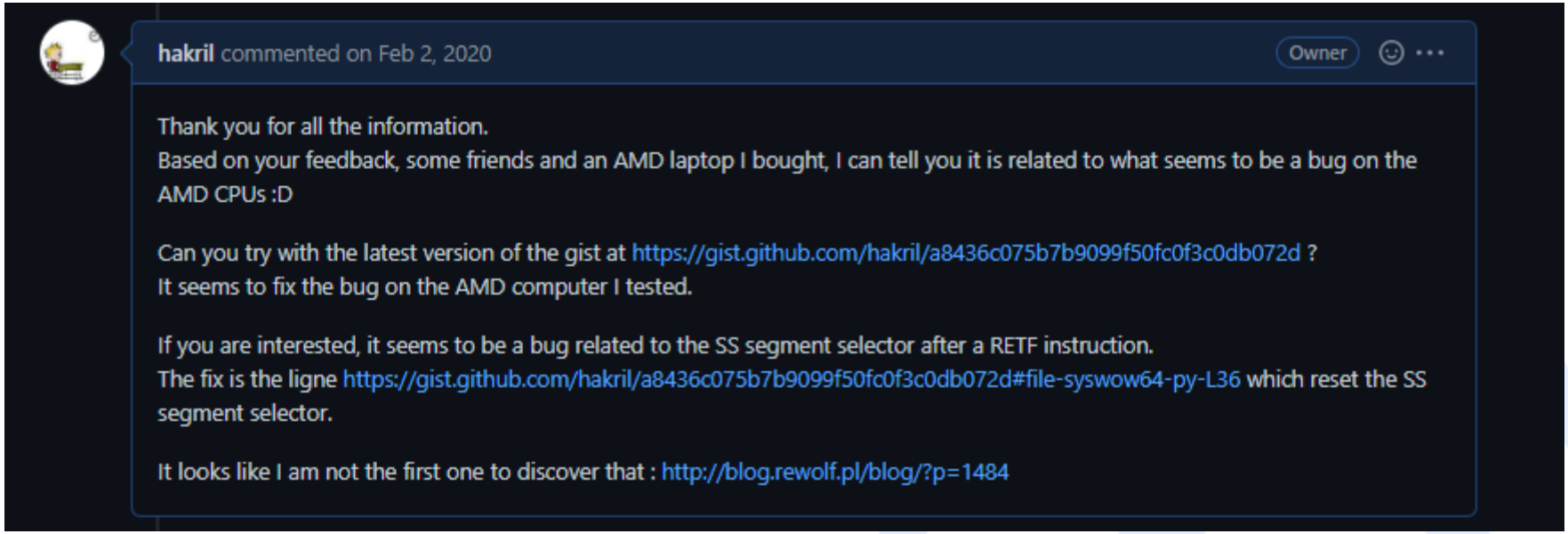
Le diff !

```
33  33
34  34      transition =    x86.MultipleInstr()
35  35      transition +=   x86.Call(CS_64bits, x64shellcodeaddr)
        36  +   # Reset the SS segment selector.
        37  +   # We need to do that due to a bug in AMD CPUs with RETF & SS
        38  +   # https://github.com/hakril/PythonForWindows/issues/10
        39  +   # http://blog.rewolf.pl/blog/?p=1484
        40  +   transition +=   x86.Mov("ECX", "SS")
        41  +   transition +=   x86.Mov("SS", "ECX")
36  42      transition +=   x86.Ret()
37  43
38  44      stubaddr = windows.current_process.allocator.write_code(transition.get_code())
```

Je n'étais pas le premier à le découvrir

- http://blog.rewolf.pl/blog/?p=1484
  - Aussi dans le cadre de travaux sur Wow64 ☺

Fin de l’issue !



hakril commented on Feb 2, 2020 — Owner

Thank you for all the information.
Based on your feedback, some friends and an AMD laptop I bought, I can tell you it is related to what seems to be a bug on the AMD CPUs :D

Can you try with the latest version of the gist at https://gist.github.com/hakril/a8436c075b7b9099f50fc0f3c0db072d ?
It seems to fix the bug on the AMD computer I tested.

If you are interested, it seems to be a bug related to the SS segment selector after a RETF instruction.
The fix is the ligne https://gist.github.com/hakril/a8436c075b7b9099f50fc0f3c0db072d#file-syswow64-py-L36 which reset the SS segment selector.

It looks like I am not the first one to discover that : http://blog.rewolf.pl/blog/?p=1484

Jamais eu de réponse ☹

Question ?

- Clement Rouault (@hakril)
- https://github.com/hakril/pythonforwindows
- https://exatrack.com