find_library considered harmful

What went wrong?

The following text holds only for Windows.

I have written a program for a DSO (that is a digital storage oscilloscope for software only geeks).

This device needs a library to enable communication with userland software.

This library must be installed before the program can be used. I call this the installed library.

The library used was libusb0 installed by inf-wizard.

Between the library and me is a software layer pyusb1. This package uses the find_library utility to select a library for my device at run time.

Unfortunately in a client's environment this procedure selected a library that ran havoc.

I call it the wrong library because it is not the installed one.

I have reproduced this failure and will demonstrate how the same library will work if installed and fail if not. See the protocols to prove it at the end.

There was nothing wrong with my client's library, it only was not installed for the DSO.

The truth is:

pyusb1 is broken because of the find_library search.

Any program works only if the used library is the installed one.

There may be exceptions to this rule, but this does not matter.

This statement has been verified for only a few cases, but that is enough.

Software must never fail for systematic errors.

The goal of this activity is to get the authors of pyusb1 to abandon the library search in favor of an installed library search.

The installed library

If you look at the control panel for the DSO device you will eventually get at the driver file details page.

Windows knows which are the installed libraries and the control panel can display them.

What we need is to get at this information. It is not in the registry (what I erroneously assumed)

but it is available.

The end user has made a distinct decision about the library he/she wants to be used for his DSO device, I hope a well founded. It is impolite, if not insulting or offending, to ignore this decision.

And it is erroneous.

What were the results of my complaints?

The authors have taken a great effort to clarify my errors and show solutions for my problem.

I want to thank them heartily.

However they only offered workarounds to enforce the use of a desired library.

Nobody questioned the inherent flaw of the find_library search.

It is time to be honest.

The find_library search must be disabled. The installed library must be found and used.

What can everybody do?

The statement about the need for the installed library should be hardened.

Everybody who has a device under control of pyusb1 can do the following:

1. Hide the library used but offer a different selectable one
2. Deinstall the libraries with the control panel and check the delete option,
   but keep the libraries available
3. Report the results in the mailing list.

I hope that many users will support my initiative to make pyusb1 a tool with an excellent reputation for reliability, ease of use, and functionality.

Hermann Hamann 2016-08-23

Protocols

This is the program to show the failure:

```
from __future__ import print_function
import platform, sys,os,ctypes
import traceback
try:
    arg1 = sys.argv[1]
except: arg1 = None

if arg1:
    print ("architecture  ",platform.architecture())

    print ("python version",platform.python_version())
    s = platform.system()
    print ("system      ",s)
    s = s[0]

    print ("win32 version ",platform.win32_ver())
    print ("64 bit system ",sys.maxsize > 2**32)
    version = sys.version_info
#print ("version",version[0],version[1])

os.environ['PYUSB_DEBUG']= "debug"

try:
```

```
  import usb
  import usb.core
  try:
    usbversion = usb.__version__
  except:
    usbversion = "?"
  print("usb", usbversion,"is available\n")
except Exception as message:
  print("usb is not available",message)
  sys.exit()

try:
  dev = usb.core.find(idVendor=0x5656)
  print ("found Product 0x%04x" % (dev.idProduct,),"on /dev/bus/usb/%03i/%03i" %
(dev.bus,dev.address))
except Exception as message:
  print ("find Exception",message)
  traceback.print_exc()
  sys.exit()


try:
  dev.set_configuration()
  response= dev.ctrl_transfer(0xc2,0xb2,wIndex=4,wValue=0x08,timeout=300,data_or_wLength=8)
  print("response",response)
except Exception as message:
  print("no response",message)
  traceback.print_exc()
```

```
  sys.exit()
try:
  dev.ctrl_transfer(0x42,0xb1,0x2c,0,None,timeout=3000)
  print("ctrl_transfer succeeded\ninspection succeeded")
except Exception as message:
  print("ctrl_transfer failed",message,"\n")
  traceback.print_exc()
```

This is the used configuration:

```
architecture   ('32bit', 'WindowsPE')
python version 2.7.12
system       Windows
win32 version  ('7', '6.1.7601', 'SP1', u'Multiprocessor Free')
64 bit system  False
```

This is the inspection result with following scenario: libusb0 is visible but not installed

```
C:\Users\ich\wdso>python inspect.py
usb 1.0.0 is available

2016-08-22 13:10:04,464 INFO:usb.libloader:candidate usb-1.0.dll
2016-08-22 13:10:04,464 INFO:usb.libloader:candidate libusb-1.0.dll
2016-08-22 13:10:04,464 INFO:usb.libloader:candidate usb.dll
2016-08-22 13:10:04,464 ERROR:usb.libloader:'Libusb 1' could not be found
2016-08-22 13:10:04,464 ERROR:usb.backend.libusb1:Error loading libusb 1.0 backend
2016-08-22 13:10:04,464 INFO:usb.libloader:candidate openusb.dll
```

2016-08-22 13:10:04,480 ERROR:usb.libloader:'OpenUSB library' could not be found

2016-08-22 13:10:04,480 ERROR:usb.backend.openusb:Error loading OpenUSB backend

2016-08-22 13:10:04,480 INFO:usb.libloader:candidate usb-0.1.dll

2016-08-22 13:10:04,480 INFO:usb.libloader:candidate usb.dll

2016-08-22 13:10:04,480 INFO:usb.libloader:candidate libusb0.dll

2016-08-22 13:10:04,480 INFO:usb.libloader:loaded C:\Windows\system32\libusb0.dll

2016-08-22 13:10:04,496 INFO:usb.core:find(): using backend "usb.backend.libusb0"

2016-08-22 13:10:04,496 DEBUG:usb.backend.libusb0:_LibUSB.enumerate_devices()

find Exception 'NoneType' object has no attribute 'idProduct'

Traceback (most recent call last):

  File "inspect.py", line 37, in <module>

    print ("found Product 0x%04x" % (dev.idProduct,),"on /dev/bus/usb/%03i/%03i" % (dev.bus,dev.address))

AttributeError: 'NoneType' object has no attribute 'idProduct'

This is the result of the following scenario:

libusb0 is installed

C:\Users\ich\wdso>python inspect.py

usb 1.0.0 is available

2016-08-22 13:16:17,723 INFO:usb.libloader:candidate usb-1.0.dll

2016-08-22 13:16:17,723 INFO:usb.libloader:candidate libusb-1.0.dll

2016-08-22 13:16:17,723 INFO:usb.libloader:candidate usb.dll

2016-08-22 13:16:17,723 ERROR:usb.libloader:'Libusb 1' could not be found

2016-08-22 13:16:17,723 ERROR:usb.backend.libusb1:Error loading libusb 1.0 backend

2016-08-22 13:16:17,723 INFO:usb.libloader:candidate openusb.dll

2016-08-22 13:16:17,739 ERROR:usb.libloader:'OpenUSB library' could not be found

2016-08-22 13:16:17,739 ERROR:usb.backend.openusb:Error loading OpenUSB backend

2016-08-22 13:16:17,739 INFO:usb.libloader:candidate usb-0.1.dll

2016-08-22 13:16:17,739 INFO:usb.libloader:candidate usb.dll

2016-08-22 13:16:17,739 INFO:usb.libloader:candidate libusb0.dll

2016-08-22 13:16:17,739 INFO:usb.libloader:loaded C:\Windows\system32\libusb0.dll

2016-08-22 13:16:17,739 INFO:usb.core:find(): using backend "usb.backend.libusb0"

2016-08-22 13:16:17,753 DEBUG:usb.backend.libusb0:_LibUSB.enumerate_devices()

2016-08-22 13:16:17,753
DEBUG:usb.backend.libusb0:_LibUSB.get_device_descriptor(<usb.backend.libusb0._usb_device object

at 0x024E3D50>)

found Product 0x0834 on /dev/bus/usb/000/001

2016-08-22 13:16:17,753
DEBUG:usb.backend.libusb0:_LibUSB.get_configuration_descriptor(<usb.backend.libusb0._usb_device

object at 0x024E3D50>, 0)

2016-08-22 13:16:17,753 DEBUG:usb.backend.libusb0:_LibUSB.open_device(<usb.backend.libusb0._usb_device object at 0x024E3

D50>)

2016-08-22 13:16:17,769 DEBUG:usb.backend.libusb0:_LibUSB.set_configuration(7345088, 1)

2016-08-22 13:16:17,769 DEBUG:usb.backend.libusb0:_LibUSB.ctrl_transfer(7345088, 194, 178, 8, 4, array('B',

[0, 0, 0, 0, 0, 0, 0, 0]), 300)

response array('B', [204, 143, 93, 218, 174, 155, 71, 123])

2016-08-22 13:16:17,801 DEBUG:usb.backend.libusb0:_LibUSB.ctrl_transfer(7345088, 66, 177, 44, 0, array('B'), 3000)

ctrl_transfer succeeded

inspection succeeded

2016-08-22 13:16:17,848 DEBUG:usb.backend.libusb0:_LibUSB.close_device(7345088)

These are the results of the following scenario:

libusb0 is visible and installed

libusb-1.0 is visible but not installed

C:\Users\ich\wdso>python inspect.py

usb 1.0.0 is available

2016-08-22 15:48:16,276 INFO:usb.libloader:candidate usb-1.0.dll

2016-08-22 15:48:16,276 INFO:usb.libloader:candidate libusb-1.0.dll

2016-08-22 15:48:16,276 INFO:usb.libloader:loaded C:\Windows\system32\libusb-1.0.dll

2016-08-22 15:48:16,292 DEBUG:usb.backend.libusb1:_LibUSB.__init__(<WinDLL 'C:\Windows\system32\libusb-1.0.dll', handle

73ef0000 at 25009f0>)

2016-08-22 15:48:16,384 INFO:usb.core:find(): using backend "usb.backend.libusb1"

2016-08-22 15:48:16,384 DEBUG:usb.backend.libusb1:_LibUSB.enumerate_devices()

2016-08-22 15:48:16,431
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551EF0>)

2016-08-22 15:48:16,431
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551FD0>)

2016-08-22 15:48:16,447
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551FF0>)

2016-08-22 15:48:16,447
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551F10>)

2016-08-22 15:48:16,447
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551FB0>)

2016-08-22 15:48:16,463
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551F30>)

2016-08-22 15:48:16,463

DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551FD0>)

2016-08-22 15:48:16,463
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551EF0>)

2016-08-22 15:48:16,463
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551F10>)

2016-08-22 15:48:16,479
DEBUG:usb.backend.libusb1:_LibUSB.get_device_descriptor(<usb.backend.libusb1._Device object at 0

x02551FF0>)

found Product 0x0834 on /dev/bus/usb/003/003

2016-08-22 15:48:16,479
DEBUG:usb.backend.libusb1:_LibUSB.get_configuration_descriptor(<usb.backend.libusb1._Device obje

ct at 0x02551FF0>, 0)

2016-08-22 15:48:16,479 DEBUG:usb.backend.libusb1:_LibUSB.open_device(<usb.backend.libusb1._Device object at 0x02551FF0>

)

2016-08-22 15:48:16,493
DEBUG:usb.backend.libusb1:_LibUSB.set_configuration(<usb.backend.libusb1._DeviceHandle object at

0x024738F0>, 1)

2016-08-22 15:48:16,493 DEBUG:usb.backend.libusb1:_LibUSB.ctrl_transfer(<usb.backend.libusb1._DeviceHandle object at 0x0

24738F0>, 194, 178, 8, 4, array('B', [0, 0, 0, 0, 0, 0, 0, 0]), 300)

response array('B', [204, 143, 95, 218, 47, 255, 119, 255])

2016-08-22 15:48:16,540 DEBUG:usb.backend.libusb1:_LibUSB.ctrl_transfer(<usb.backend.libusb1._DeviceHandle object at 0x0

24738F0>, 66, 177, 44, 0, array('B'), 3000)

ctrl_transfer failed [Errno 5] Input/Output Error

Traceback (most recent call last):

File "inspect.py", line 53, in <module>

dev.ctrl_transfer(0x42,0xb1,0x2c,0,None,timeout=3000)

File "C:\Python27\lib\site-packages\usb\core.py", line 1043, in ctrl_transfer

self.__get_timeout(timeout))

File "C:\Python27\lib\site-packages\usb_debug.py", line 60, in do_trace

return f(*args, **named_args)

File "C:\Python27\lib\site-packages\usb\backend\libusb1.py", line 883, in ctrl_transfer

timeout))

File "C:\Python27\lib\site-packages\usb\backend\libusb1.py", line 595, in _check

raise USBError(_strerror(ret), ret, _libusb_errno[ret])

USBError: [Errno 5] Input/Output Error

2016-08-22 15:48:16,618 DEBUG:usb.backend.libusb1:_LibUSB.close_device(<usb.backend.libusb1._DeviceHandle object at 0x02

4738F0>)

2016-08-22 15:48:16,618 DEBUG:usb.backend.libusb1:_LibUSB._finalize_object()

Note: the indicated bus/device looks wrong.

The indicated path if wrong, it should be Windows/syswow64/

There was no timeout. The first control transfer succeeded.

The output with libusb-1.0 installed is missing because I could not get the inf-file working. Sorry.

But this library works at my client for the installed device.