# Writeup for phase 1

Jiahao Cai

Oct 5, 2018

- **0. Extract Python code from pdf file, according to the hint from the suffix of my resume (.py.pdf).**

- **1. Try to print the type of everything to get some hints, e.g.**
  - code object
  - the filename is called `russian_doll.py`

- **2. Start to know `marshall` module, try to use if I can get source code from it.**

- **3. Try to save the decompressed content as `.zip`, and unzip it. Then realize it is endless, it corresponds to the name `russian_doll.py`.**

- **4. Drag zip file into text editor, realize it is a Python file indeed, realize that this is the file which will be executed later.**

- **5. Add exception to see that what really happened. It turns out that it will do something to get the correct marshall data according to the input. Then I realize what the hint means, the first input should be `n`.**

```
try:
    c1(x, y)
except Exception, e:
  print "Exception: " + str(e)
```

•

**6. Print more things to see what happened, then realize what happened, the length of the answer should be 13.**

```
def c1(x, y):
    print "New c1..."
    eval(marshal.loads(x))
    print base64.b64decode(y) # Bunch of encrypted code
    exec base64.b64decode(y)
    print '-' * 77
```

•

**7. Try to search for disassembler for marshall object, then I found dis. Try to disassemble the code object in the source code.**

•

**8. First I disassembled x_func(), which you can see it here, important thing is there is a time interval check, which is very interesting.**

```
(Pdb) p x.co_varnames
('message', 'split_string', 't0', 'fd', 'old_settings', 'res', 't1', 'x', '_')
(Pdb) p x.co_consts
(None, ':R:', 1, '',
<code object <lambda> at 0x100a712b0, file "/usr/local/bin/russian_doll.py", line 44>,
4,
<code object <lambda> at 0x101804030, file "/usr/local/bin/russian_doll.py", line 47>,
'finished', 0)
(Pdb) p x.co_cellvars
('passphrase',)
34           0 LOAD_CONST               1 (':R:')
             3 STORE_FAST               1 (split_string)

36           6 LOAD_GLOBAL              0 (time)
             9 LOAD_ATTR                0 (time)
            12 CALL_FUNCTION            0
            15 STORE_FAST               2 (t0)
```

```
......

40          70 LOAD_GLOBAL              8 (ord)
            73 LOAD_GLOBAL              1 (sys)
            76 LOAD_ATTR                2 (stdin)
            79 LOAD_ATTR                9 (read)
            82 LOAD_CONST               2 (1)
            85 CALL_FUNCTION            1
            88 CALL_FUNCTION            1
            91 STORE_DEREF              0 (passphrase) # passphrase = ord(input)

42          94 LOAD_GLOBAL              4 (termios)
            97 LOAD_ATTR               10 (tcsetattr)
            100 LOAD_FAST               3 (fd)
            103 LOAD_GLOBAL             4 (termios)
            106 LOAD_ATTR              11 (TCSADRAIN)
            109 LOAD_FAST               4 (old_settings)
            112 CALL_FUNCTION           3
            115 POP_TOP

44         116 LOAD_CONST               3 ('')
            119 LOAD_ATTR              12 (join)
            122 LOAD_GLOBAL            13 (map)
            125 LOAD_CLOSURE            0 (passphrase)
            128 BUILD_TUPLE             1 # ('').join, map, res)
            131 LOAD_CONST              4 (<code object <lambda> at 0x10476d7b0,
            file "/usr/local/bin/russian_doll.py", line 44>) # return value of code
            134 MAKE_CLOSURE            0
            137 LOAD_FAST               0 (message)
            140 CALL_FUNCTION           2
            143 CALL_FUNCTION           1
            146 STORE_FAST              5 (res)

45         149 LOAD_GLOBAL              0 (time)
            152 LOAD_ATTR               0 (time)
            155 CALL_FUNCTION           0
            158 STORE_FAST              6 (t1) # t1 = now()

46         161 LOAD_FAST                6 (t1)
            164 LOAD_FAST               2 (t0)
            167 BINARY_SUBTRACT               # t = t1 - t2
            168 LOAD_CONST              5 (4) # t > 4
            171 COMPARE_OP              4 (>)
            174 POP_JUMP_IF_FALSE     213     # if (t < 4) goto wrong
```

```
47          177 LOAD_CONST               3 ('')
             180 LOAD_ATTR               12 (join)
             183 LOAD_GLOBAL             13 (map)
             186 LOAD_CLOSURE             0 (passphrase)
             189 BUILD_TUPLE              1 # (''.join.map, passphrase)
             192 LOAD_CONST               6 (<code object <lambda> at 0x10476d830,
             file "/usr/local/bin/russian_doll.py", line 47>)
             195 MAKE_CLOSURE             0
             198 LOAD_FAST                0 (message)
             201 CALL_FUNCTION            2
             204 CALL_FUNCTION            1
             207 STORE_FAST               5 (res)
             210 JUMP_FORWARD             0 (to 213)

48      >>  213 LOAD_FAST                5 (res)
             216 LOAD_ATTR               14 (split)
             219 LOAD_FAST                1 (split_string) # :R:
             222 CALL_FUNCTION            1  # x = res.split(':R:')
             225 STORE_FAST               7 (x)

49          228 LOAD_GLOBAL             15 (len)
             231 LOAD_FAST                7 (x)
             234 CALL_FUNCTION            1
             237 LOAD_CONST               2 (1)
             240 COMPARE_OP               2 (==) # if (len(x) != 1) goto 266
             243 POP_JUMP_IF_FALSE      266

50          246 LOAD_CONST               7 ('finished')
             249 STORE_FAST               8 (_)

51          252 LOAD_FAST                7 (x)
             255 LOAD_CONST               8 (0)
             258 BINARY_SUBSCR                    # x[0]
             259 LOAD_FAST                8 (_)   # finished
             262 BUILD_TUPLE              2       # (x[0], 'finished')
             265 RETURN_VALUE                     # correct return

52      >>  266 LOAD_FAST                7 (x)
             269 LOAD_CONST               8 (0)
             272 BINARY_SUBSCR                    # x[0]
             273 LOAD_FAST                7 (x)
             276 LOAD_CONST               2 (1)
             279 BINARY_SUBSCR
             280 BUILD_TUPLE              2
             283 RETURN_VALUE                     # wrong return
```

-

**9. Then I add disassembler to `c1(x, y)`, in this case, everytime it use `exec` to enter a new level, it will print the disassembly code automatically:**

```
def c1(x, y):
    print "New c1..."
    eval(marshal.loads(x))
    print dis.dis(marshal.loads(x))
    exec base64.b64decode(y)
    print '-' * 77
```

-

**10. The ACSII code of the current answer is hidden in the return value of the code**

```
......
27          24 LOAD_FAST                0 (time)
30          27 LOAD_ATTR                0 (time)
33          30 CALL_FUNCTION            0
36          33 STORE_FAST               2 (_0) # current time
39          36 LOAD_CONST               2 (1539268928.71756) # October 11, 2018 2:42:0
42          39 LOAD_FAST                2 (_0) # current time
45          42 BINARY_SUBTRACT          # 1539268928.71756 - current time
46          43 STORE_FAST               3 (_2) # 1539268928.71756 - current time
49          46 LOAD_FAST                3 (_2)
52          49 LOAD_CONST               3 (0)
55          52 COMPARE_OP               0 (<) # 1539268928.71756 - current time < 0?
58          55 DUP_TOP
59          56 POP_JUMP_IF_TRUE       113     # expired
62          59 LOAD_FAST                0 (time)
65          62 LOAD_ATTR                0 (time)
68          65 CALL_FUNCTION            0
71          68 STORE_FAST               2 (_0) # current time
74          71 LOAD_FAST                0 (time)
77          74 LOAD_ATTR                0 (time)
80          77 CALL_FUNCTION            0      # current time
83          80 STORE_FAST               4 (_1)
86          83 LOAD_FAST                4 (_1)
89          86 LOAD_FAST                2 (_0)
92          89 BINARY_SUBTRACT
93          90 STORE_FAST               3 (_2) # time difference
96          93 LOAD_FAST                3 (_2)
99          96 LOAD_CONST               4 (4.1)
```

```
102          99 COMPARE_OP               4 (>) # if > 4.1s
105         102 DUP_TOP
106         103 POP_JUMP_IF_TRUE       110
109         106 LOAD_CONST               5 (118) # Answer to the next level!
112         109 RETURN_VALUE
113     >>  110 JUMP_ABSOLUTE          110
116     >>  113 LOAD_CONST               6 ('too slooooo00w!!')
119         116 PRINT_ITEM
```

-

**11. Finally we can get the answer:**

```
#  n   e    r   v   o  (t) (r) (X) (n) (") (M) (F) (})
# 110 101 114 118 111 116 114 89 110  35  77  70  125

#                        _^_                                          _^_
#                       (   )________________________________________(   )
#                       |   |                                        |   |
#                       |   |        N                               |   |
#                       |   |        |                               |   |
#                       |   |      \   /     N 40&#xb0; 45' 16.984"  |   |
#                       |   |   W -  o  - E                          |   |
#                       |   |      /   \     W 73&#xb0; 59' 38.033"  |   |
#                       |   |        |                               |   |
#                       |   |        S                               |   |
#                       (_ _)________________________________________(_ _)
#                         v                                            v

# Congrutlations! You solved Phase 1 of the problem. Get the phase 2 challenge here:
# http://www.redballoonsecurity.com/1NPTMESGGL/JFS7BSFB23.tar.gz.gpg . Key to unlock
# the challenge:'rbssecretcongratz!'. Look in my memory for further instructions.
```

-

**12. Use `lldb --attach-pid [pid]` to attach to the process and use `process save-core "core"` to dump the memory and search for the email: *o4yr7w3jo9@redballoonsecurity.com*.**

**It is a very interesting program, I have to admit that I didn't get all the details though I got the answer. When doing this challenge, I can feel that *Red Balloon Security* must be a very interesing company, and full of the old style hacker atmosphere, that's really COOL!**

**-Jiahao**