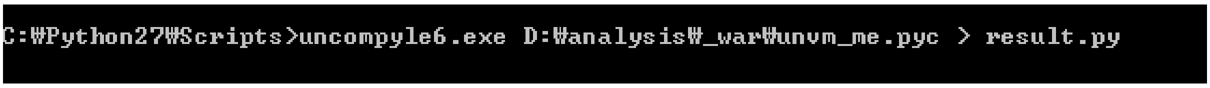
AlexCTF 2017

unvm me Write Up

Author : pandakim2122

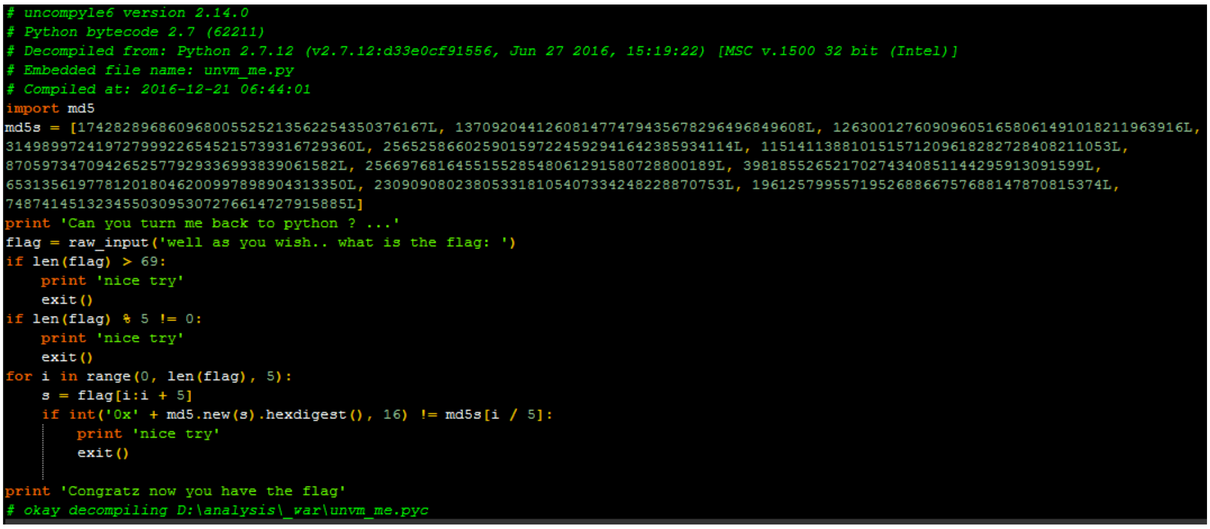
Date : 2017.12.05

umcompyle 모듈을 이용해 unvm_me.pyc 파일을 디컴파일합니다.



```
C:₩Python27₩Scripts>uncompyle6.exe D:₩analysis₩_war₩unvm_me.pyc > result.py
```

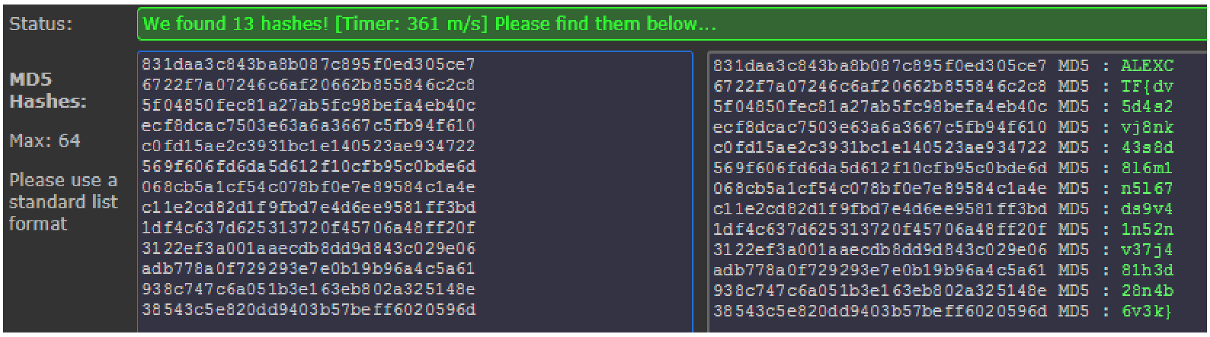
해당 파이썬 파일에는 10진수로 된 md5값들이 있는데, 입력받은 값을 5바이트 단위로 끊은 md5 값들을 이 md5 리스트와 비교합니다.



```
# uncompyle6 version 2.14.0
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)]
# Embedded file name: unvm_me.py
# Compiled at: 2016-12-21 06:44:01
import md5
md5s = [174282896860968005525213562254350376167L, 137092044126081477479435678296496849608L, 126300127609096051658061491018211963916L,
314989972419727999226545215739316729360L, 256525866025901597224592941642385934114L, 115141138810151571209618282728408211053L,
8705973470942652577929336993839061582L, 256697681645515528548061291580728800189L, 39818552652170274340851144295913091599L,
65313561977812018046200997898904313350L, 230909080238053318105407334248228870753L, 196125799557195268866757688147870815374L,
74874145132345503095307276614727915885L]
print 'Can you turn me back to python ? ...'
flag = raw_input('well as you wish.. what is the flag: ')
if len(flag) > 69:
    print 'nice try'
    exit()
if len(flag) % 5 != 0:
    print 'nice try'
    exit()
for i in range(0, len(flag), 5):
    s = flag[i:i + 5]
    if int('0x' + md5.new(s).hexdigest(), 16) != md5s[i / 5]:
        print 'nice try'
        exit()

print 'Congratz now you have the flag'
# okay decompiling D:\analysis\_war\unvm_me.pyc
```

MD5 Decrypter 사이트에서 이 값들을 입력하면 다음과 같은 결과가 나옵니다.



We found 13 hashes! [Timer: 361 m/s] Please find them below...

```
831daa3c843ba8b087c895f0ed305ce7
6722f7a07246c6af20662b855846c2c8
5f04850fec81a27ab5fc98befa4eb40c
ecf8dcac7503e63a6a3667c5fb94f610
c0fd15ae2c3931bc1e140523ae934722
569f606fd6da5d612f10cfb95c0bde6d
068cb5a1cf54c078bf0e7e89584c1a4e
c11e2cd82d1f9fbd7e4d6ee9581ff3bd
1df4c637d625313720f45706a48ff20f
3122ef3a001aaecdb8dd9d843c029e06
adb778a0f729293e7e0b19b96a4c5a61
938c747c6a051b3e163eb802a325148e
38543c5e820dd9403b57beff6020596d
```

```
831daa3c843ba8b087c895f0ed305ce7 MD5 : ALEXC
6722f7a07246c6af20662b855846c2c8 MD5 : TF{dv
5f04850fec81a27ab5fc98befa4eb40c MD5 : 5d4s2
ecf8dcac7503e63a6a3667c5fb94f610 MD5 : vj8nk
c0fd15ae2c3931bc1e140523ae934722 MD5 : 43s8d
569f606fd6da5d612f10cfb95c0bde6d MD5 : 8l6m1
068cb5a1cf54c078bf0e7e89584c1a4e MD5 : n5l67
c11e2cd82d1f9fbd7e4d6ee9581ff3bd MD5 : ds9v4
1df4c637d625313720f45706a48ff20f MD5 : 1n52n
3122ef3a001aaecdb8dd9d843c029e06 MD5 : v37j4
adb778a0f729293e7e0b19b96a4c5a61 MD5 : 81h3d
938c747c6a051b3e163eb802a325148e MD5 : 28n4b
38543c5e820dd9403b57beff6020596d MD5 : 6v3k}
```

flag : ALEXCTF{dv5d4s2vj8nk43s8d8l6m1n5l67ds9v41n52nv37j481h3d28n4b6v3k}