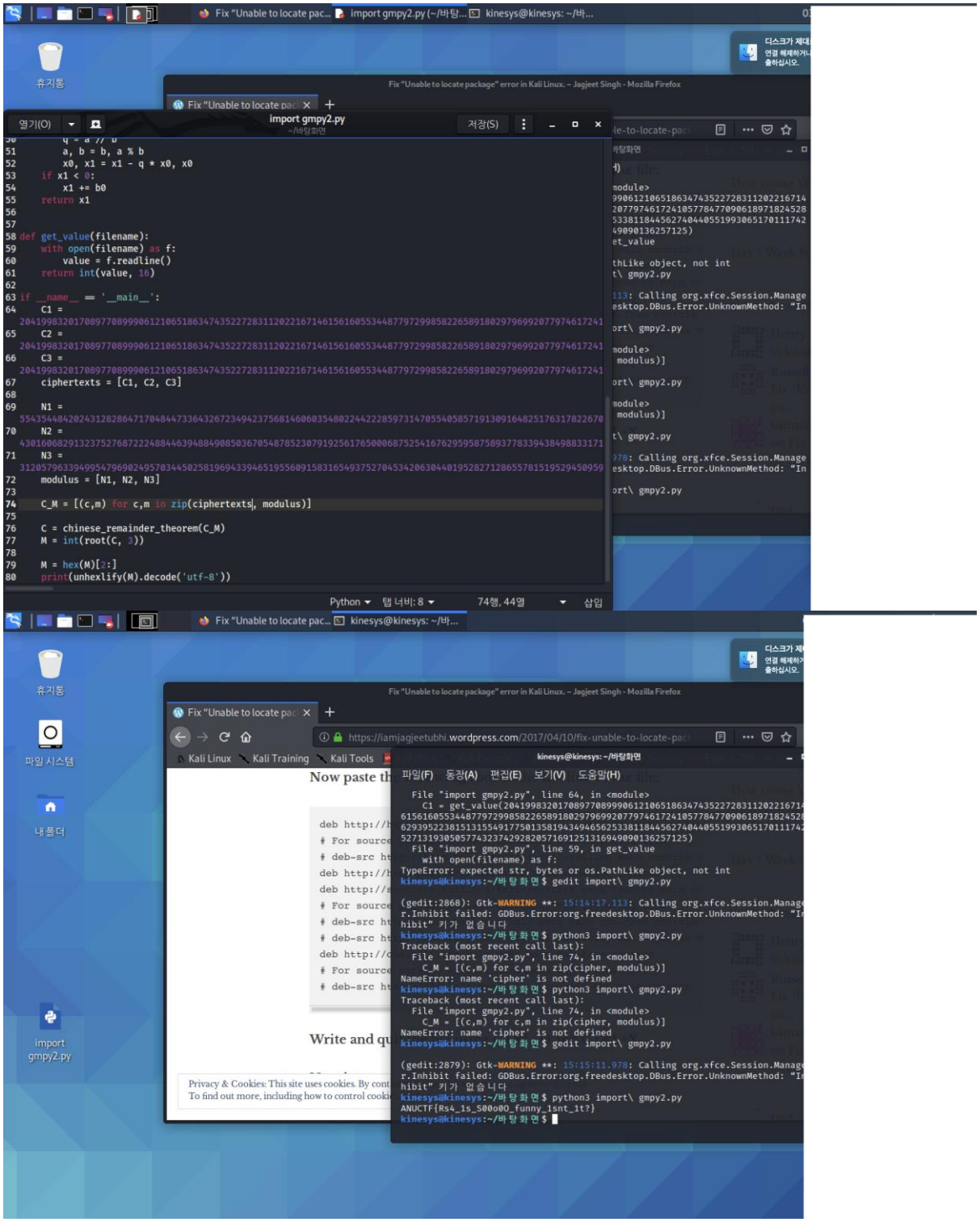
ANU CTF Writeup – moonjs

Crypto 문제의 rsa의 경우에는 하스타드 방식으로 되어있어 아래 사진과 같이 파이썬으로 익스플로잇 코드를 작성하여 실행하였더니 플래그가 나왔다

```
import gmpy2
gmpy2.get_context().precision = 4096

from binascii import unhexlify
from functools import reduce
from gmpy2 import root
import gmpy

EXPONENT = 3



def chinese_remainder_theorem(items):
    # Determine N, the product of all n_i
    N = 1
    for a, n in items:
        N *= n

    # Find the solution (mod N)
    result = 0
    for a, n in items:
        m = N // n
        r, s, d = extended_gcd(n, m)
        if d != 1:
            raise "Input not pairwise co-prime"
        result += a * s * m

    # Make sure we return the canonical solution.
    return result % N


def extended_gcd(a, b):
    x, y = 0, 1
    lastx, lasty = 1, 0

    while b:
        a, (q, b) = b, divmod(a, b)
        x, lastx = lastx - q * x, x
        y, lasty = lasty - q * y, y

    return (lastx, lasty, a)


def mul_inv(a, b):
```

```
    b0 = b
    x0, x1 = 0, 1
    if b == 1:
        return 1
    while a > 1:
        q = a // b
        a, b = b, a % b
        x0, x1 = x1 - q * x0, x0
    if x1 < 0:
        x1 += b0
    return x1


def get_value(filename):
    with open(filename) as f:
        value = f.readline()
    return int(value, 16)

if __name__ == '__main__':
    C1 =
20419983201708977089990612106518634743522728311202216714615616055344877972998582265891802979699207797461724105778477090618971824528629395223815131554917750135819434946562533811844562740440551993065170111742527131930505774323742928205716912513169490901362571250
    C2 =
20419983201708977089990612106518634743522728311202216714615616055344877972998582265891802979699207797461724105778477090618971824528629395223815131554917750135819434946562533811844562740440551993065170111742527131930505774323742928205716912513169490901362571250
    C3 =
20419983201708977089990612106518634743522728311202216714615616055344877972998582265891802979699207797461724105778477090618971824528629395223815131554917750135819434946562533811844562740440551993065170111742527131930505774323742928205716912513169490901362571250
    ciphertexts = [C1, C2, C3]

    N1 =
55435448420243128286471704844733643267234942375681460603548022442228597314705540585719130916482517631782267001157542388357579886012823986449745489383245113144052289977570100432258838339287379644060735739575165939801198091168682541741394891153053061200620079524304678969455428937416342130475667529155337915955021359228143762422934552843686419583904346036810115865850199534435714426375253256566360721864530540147430900222933950755454042927751734471251861863183994132069253077524185432486065823219419974996591043763764152246422143296684002556704213756754690168644525036990811432971601835928822709829407926039990102797724011884084889430009236279487937307217049305776753151327532103991687407539051527802782631954163841945380038452007079455473183733987239381830154964315487528354138429833064171921538923788737443962967951022767040289972605049912047050133464686980411758504674525174032814292197853148152588496306773368338395311139715830473804039782843104583389955569228162621829963699299571184726061545557761145498040981782982969747954844833965715510072662906647011386417544004587449739434823033837785061704840544940639080439455991878551125877080228103615032746378855956941001299495245962487234743296939409253145353543139108910441274485528 1
    N2 =
43016068291323752768722248844639488490850367054878523079192561765000687525416762959587589377833943849883317140460575302489768448702489929020187026962009216950338579837062206777391
```

```
840755330544298154706466821107563940929541052740437641758581421215567735330056430808607716
267829823963714994106106404517461636514934940768033727131944830974912189754178528453026115
108966860090149107897817287664480072059658505677827649938786325794441537158399916568592411
076979463144588851330764514480469357798723495286124561003041361227822483220832701754841603
047524239383365062996735965136495195685669923898613824177455927843097040157518020837700340
593910682432528253909351393125855920706309494753537972669635834625670774914163702639655784
305168032953095559080296816916664671670020315166917150037155231590244919321564154158586808
595109144208399081612551884746915156917012163601269179394962836345459014429559185573301706
358212160303882066991980804347360387291835340657770979469077495011086151391441239626297071
737404232987994954865282095243453593589786498317830872771540172298003884560025529651521850
016511150539769439512394785454888615982904803617962145122452585997235770005585083192462196
825768932992222631956306695565927542991257229370050669738614463
    N3 =
312057963394995479690249570344502581969433946519556091583165493752704534206304401952827128
655781519529450959622528130616721790483048088986052748712918739176921301033847708826280483
411077372867688770216497380310915997936605980292737372081069269295411079993739133613028980
805834705158592414996360732383877551563496274488888245158712912021966334099145777836138001
353181346735184097562254827024694074128259961495326174693109423957696219207495984195630155
011457771756976304965737821396723096727010998867671048498075153403092731663159674933154900
269959748890373800925729078630741488793670016023971925771130305659187602753690264288901722
847555254246782238047829339540780950679336683237034989517100117332806507228094062006438545
878782810651536734396044271048546623545987884676536715771603618253514090680250206990426753
299344434071469488992091240452698035144103979249335236687870291782969899354657958732943734
097117569589094053054185618828162750473415131026811578766803323402423873323863632918065660
512508906813727376272138919600007134803446657536099513549065391904811937374492315393066408
311083605969429260579510505353364119934684053963904600278328620529201613401710628415698417
250332657396854162778714055619556622784177829844738721747195 97
    modulus = [N1, N2, N3]

    C_M = [(c,m) for c,m in zip(ciphertexts, modulus)]

    C = chinese_remainder_theorem(C_M)
    M = int(root(C, 3))

    M = hex(M)[2:]
    print(unhexlify(M).decode('utf-8'))
```


Fix "Unable to locate pac...
import gmpy2.py (~/바탕...
kinesys@kinesys: ~/바...
휴지통
Fix "Unable to locate package" error in Kali Linux. – Jagjeet Singh - Mozilla Firefox
import gmpy2.py
~/바탕화면
열기(O)
저장(S)
def get_value(filename):
with open(filename) as f:
value = f.readline()
return int(value, 16)
if __name__ == '__main__':
C1 =
C2 =
C3 =
ciphertexts = [C1, C2, C3]
N1 =
N2 =
N3 =
modulus = [N1, N2, N3]
C_M = [(c,m) for c,m in zip(ciphertexts, modulus)]
C = chinese_remainder_theorem(C_M)
M = int(root(C, 3))
M = hex(M)[2:]
print(unhexlify(M).decode('utf-8'))
Python
탭 너비: 8
74행, 44열
삽입
휴지통
파일 시스템
내 폴더
import gmpy2.py
Kali Linux
Kali Training
Kali Tools
kinesys@kinesys: ~/바탕화면
파일(F) 동작(A) 편집(E) 보기(V) 도움말(H)
File "import gmpy2.py", line 64, in <module>
C1 = get_value(20419983201708977089990612106518634743522728311202216714
File "import gmpy2.py", line 59, in get_value
with open(filename) as f:
TypeError: expected str, bytes or os.PathLike object, not int
kinesys@kinesys:~/바탕화면$ gedit import\ gmpy2.py
kinesys@kinesys:~/바탕화면$ python3 import\ gmpy2.py
Traceback (most recent call last):
File "import gmpy2.py", line 74, in <module>
C_M = [(c,m) for c,m in zip(cipher, modulus)]
NameError: name 'cipher' is not defined
kinesys@kinesys:~/바탕화면$ python3 import\ gmpy2.py
Traceback (most recent call last):
File "import gmpy2.py", line 74, in <module>
C_M = [(c,m) for c,m in zip(cipher, modulus)]
NameError: name 'cipher' is not defined
kinesys@kinesys:~/바탕화면$ gedit import\ gmpy2.py
kinesys@kinesys:~/바탕화면$ python3 import\ gmpy2.py
ANUCTF{Rs4_1s_S00o0O_funny_1snt_1t?}
kinesys@kinesys:~/바탕화면$

실행환경 : 칼리리눅스