

BUGGY_TIME_MACHINE



CHALLENGE INFO

Nuclear Disaster

The country is under red alert! Hackers have compromised the computer controlling the cooling system of the reactors! They have changed the root password and without it we cannot restore the system. Fortunately, we can still access the computer with one of our employees accounts. It seems like the hackers left some things back. Could you help us prevent a nuclear disaster? huan:anothermonday9%



This challenge is started on-demand.

MATERIAL:

—

FLAG:

HTB{s3cur3_y0ur_cr1t1c4l_1nfr4s7ruc7ur3}

SOLVER:

M1gnus

FOOTHOLD

The challenge provide an username and a password and a server reachable with ssh: `ssh huan@docker.hackthebox.eu -p30265` (password anothermonday9%) .

```
huan@2da296184d7b:~$ find .
```

```
./bashrc
./bash_logout
./profile
./cache
./cache/motd.legal-displayed
./your_last_hope.txt
./secret
./secret/not_the_encryption_script.py
```

by using find is possible to spot two interesting file:
your_last_hope.txt:

```
n:
0x690ee43793bc1f34b9e44f7aeb91063d92292be0884816718ed836feec60744d68f73c0dcebe42837a7d316dd0ab8589b461d034b0e6468
e734b2b741c2ce7d50ca0e619114d758ff1973e57f843f22b0cb14af7694763a774f306c4f106616f173e2b4bdb49ef1cc2553c9dd3dc1877f
7f87f106d01052faae22acf6d6f3fbbfdb41be6192a0e49810a7c0874d634bcd07627e2a64f596e4356212f38628bb5416a6008e2ef0763a0da
97475d7d5a1ee15842a4db54987b776f36120bf851b70c7edd197857297193c39c69d3cefe330029e57baae71f57a8fa3ffe3392855d7eca38
61aa60eaabf93244b6659c9838680829eeba788dba871eedaf1f48e6ed00c1d5c12ace91edfab730d88180f2cab20beca486f73bb39ff73120
a7d80c1a4bdf6a3507009737977262a534a6513f4fb2db9a626f33d50a16f11098ada1d8f73043e616c298ce5bcc657120b83daa7e35263d0f
f9dae602df9e00b1da0b0beacb00ee680ca890f8949f9ebd723c81eb789309a007301e6a621d431d367577

e:
0x257ae0a7b65e25b4e6de59c94259c86238666858419de8565c6bfc5ce2c7964bf013918b888341361ab61143a3848e34c8609b8eed9f44bb
ad6ec5c2e9c5d1fdcf79038a1fcb87aad9cae2cd885e2f42758b3ccc1efd37ebddc4462f86af68c199ae6015600077ddf6b8e2dfa5ce5b9701
aebbd2d1385171c300a2823d72511f5663692b0c33c7d5ccdc9acd255a0ef7d644d94670a7af40a6829494e8a8371657b8b96618b432303071
3b32c29b159beddd7a0af42c64c63b5d92a8b8ff493b0418828206fbb5663d35d7d42d03e54b37e08f858296234d85d1eb72ad6fdff7115799
c4901635cfd2ac83dda30b69fd9fbb9d1bddf38fd9c62e01776ab3c155d0e3fecefb7b9801e0cbc477b10dcd25f872c0f3172cdf05f27a17ea
6d418f77bec591e1828b3830d35a3217cf59c7c384eee460579ff5d61f368bf453b142952456db7934d0063c0f46d9b5a8b35a7e7a4b211c5
32b7ce7ca5c09e45e241f87eaf63e0ecd35ece6275509c999415138daa6530297564e32b298dda62d87bf

a:
0xd05e30b62b9c8a473b7513ac2c8b372d58092733a463458f749b2db4544a868344f79312dd03c366f6855f12db68881d3c57ba857c8c4927
a93e8e59569ba1c8eada720adcc818b8fb92c3e0b23ab26704d7859c16ef09c9d2692f0132295fa8a67e7936d1a7d53872c3a51e7115cb952c
3cbbbf8bd29c2f5b58ecdb799b423ebeca3725ff317682750610e5986b9c0fb385feacdbfda4082b8e820a7651e9569642d598c5005017601
9ade5ef446b5165b8280c0f0c3f33efed0d13298aa6ab628669493d12bbbb6e80bc220b303874f0b4acd8a61319215fb44decae0c2f116e0710
d3b022baeb6ee294206310e42dd51c1de91de872f1adf5c395a674cf05

g:
0x23e774b678534aca7671d71a4f87dca2274e161b1f8561ab2ad51d12333d901f054759281a8bc1f9686184b5487e8353813dc0938e1570ae
d83f34857744fbc5ea856687dcac06c53e3f9fc7e973cc94efb17ba85ecc00109e289b157172a0cad89ef4b6d795271e4e42aca26f0a0ee3cd
71e2f561657fca7ddcb0fb45bb4d0491dca103cb5a4bfbbbf086bb58f73ae4f0fbd5fd8b2baf3606aca6937f0cd4a764be6ab44a014c6be5d2
614a9de6c6548ce86588b7a3cff8e326429c9c2dcdd8595d9c5703f91c07b835721d5b59d9ba26e3f4d34f7e38a2d39602259cb024da307715
f61a0c07ea4ce47eb19356109e4335820faa5b6e1f68fdcd4780c92a1c

c1:
0x48eeea8b6d8ee7b7264115858c883005b93a515db96c33329f22153b5646d771aee09fd5e34338640ddd3bb933e1eda111625f527725d3e3
b8a7990560a4a50271fce090fcd1f525a9ddb5ceac20646afa5d383a84eab00c88fb6041899ebc1a6ce0b10a2fe00a7b6f7c0a23d6bb4c0223
7a4eaf7a8b8a748e1273a873d8b0b48526a6b96675163bd20d452050d530bd2cad6849a7087663a3a3d11a2063716e4bd0279f1c369538d49
1b891301126310abb5b3b2ed1c2d4547fec0feba29fala14b641e2cb5669cc6225a0f9c573ef3dd2a4b8f820afe63a639538d4dc52c1b7638
0ed3f6e9507c40f1a68412585acea0e5a8b5f00fb15187ed84a65db1dba5e2c2a8234cc3acb15b354c3072e5f9e330f12aa6a323e4f981850
50a161621fcaa6a0067697420c5a9509ed35724a623e36963fa80a5831d6925b2a89be18935147cfc2d390cbe39f3e44a4951f3a6c078c10d2
c241c6e9a1e84188b36abee3c045d61325ba73b12c54844a2debfb93d2d5374679f9aaa668278893e0ef65c

c2:
0xb9aa388ec95aa402d590fd5d7f5ccb8cc1c741060cf4a33147ff5b030b8ab160d7475befb59a5221ab48a4567977f40ce5a60ec32ed35d21f0
5f3d9a3c4f0cc129f8bb71193cd14752e9b446132fca31402fc38d32b41b5cd98b8d56b03e360ab778158f2b529b95adcef7014fb7914958f7
3eb866232d58c06034c9050e94dd47bca4b43b7d3c9990171b6bfced26ec22779a229cbd5d89b46110bfb17efcd68a6f786502a3e3e94dfac1
1b30108109ed331d2b3d69a70716d48420f0c1b43e954a019db1bf8a47f73360db789df6fd5ae360fc6d2d5952e41f1ce4def3ce2454007711
c7c0fd4359c30fea7357e6ec1c5e61a42e908c0d01cb145795b44bf505

DROP US some ETH and you may prevent a nuclear disaster:
0x7b1cA37A0ad47B14e55a1E0d9d882999c0DF1Ee0
```

```
./secret/not_the_encryption_script.py:
```

```
from Crypto.Util.number import *
from secret import root

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

def generate_keys(nbit):
    p, q, r = [getPrime(nbit) for _ in range(3)]

    n = p * q * r

    phi = (p-1)*(q-1)*(r-1)

    d = getPrime(1 << 8)
    e = modinv(d, phi)
    a = getPrime(2*nbit)
    while True:
        g = getRandomRange(2, a)
        if pow(g, 2, a) != 1 and pow(g, a//2, a) != 1:
            break

    pub_key = (n, e, a, g)

    priv_key = (n, d, a, g)

    return pub_key, priv_key

def encrypt(m, pub_key):
    n, e, a, g = pub_key
    k = getRandomRange(2, a)
    K = pow(g, k, a)
    c1, c2 = pow(k, e, n), (m * K) % a
    return c1, c2

password = bytes_to_long(root)
pub_key, priv_key = generate_keys(1024)

c1, c2 = encrypt(password, pub_key)

f = open('your_last_hope.txt', 'w')
f.write('n: ' + hex(pub_key[0]) + '\n')
f.write('e: ' + hex(pub_key[1]) + '\n')
f.write('a: ' + hex(pub_key[2]) + '\n')
f.write('g: ' + hex(pub_key[3]) + '\n')

f.write('c1: ' + hex(c1) + '\n')
f.write('c2: ' + hex(c2) + '\n')
f.write('DROP US some ETH if you know what\'s good:\n0x7b1cA37A0ad47B14e55a1E0d9d882999c0DF1Ee0\n')
f.close()
```

The reasonable suspect is that, despite its name, the second file is the script used to generate the first file. Is possible to acquire more informations by analyzing the code. The encrypted data is the root password, the encryption protocol is clear, we have a [RSA](#)-like couple of exponent (one public and the other private) and a series of modular operations. Once d is obtained, the secret message can be easily recovered:

```
k = pow(c1, d, n)
m = (c2 * inverse_mod(pow(g, k, a)))%a
```

THE ATTACK

The private exponent d is generated really smaller than n , this means that maybe a [Wiener factorization attack](#) should be possible, but n is not the product of two primes, but is the product of three primes, this seems to be bad since the equations exploited by Wiener is made for numbers that are product of two primes. Luckily Wiener calculate ϕ to perform the factorization, and that's exactly what we need to recover d . So the only thing to do is to make some changes to Wiener factorization attack, recover the correct ϕ and obtain the flag.

*In the classic attack Wiener calculate many ϕ 's and tries to recover p and q from any of them, when $p*q == n$ Wiener ends the attack and return the results. Is possible to simply encrypt something with n , e and test the goodness of ϕ by calculating d and recover the message.

THE IMPLEMENTATION

```
import binascii
```

```
n =
0x690ee43793bc1f34b9e44f7aeb91063d92292be0884816718ed836feec60744d68f73c0dcebe42837a7d316dd0ab8589b461d034b0e6468e734b2b741c2ce7d50ca0e61911
4d758ff1973e57f843f22b0cb14af7694763a774f306c4f106616f173e2b4bdb49ef1cc2553c9dd3dc1877f7f87f106d01052faae22acfd6f3fbbfdb41be6192a0e49810a7c0874d
634bcd07627e2a64f596e4356212f38628bb5416a6008e2ef0763a0da97475d7d5a1ee15842a4db54987b776f36120bf851b70c7edd197857297193c39c69d3cefe330029e57b
aae71f57a8fa3ffe3392855d7eca3861aa60eaabf93244b6659c9838680829eeba788dba871eedaf1f48e6ed00c1d5c12ace91edfab730d88180f2cab20beca486f73bb39ff73120
a7d80c1a4bdfea3507009737977262a534a6513f4fb2db9a626f33d50a16f11098ada1d8f73043e616c298ce5bcc657120b83daa7e35263d0ff9dae602df9e00b1da0b0beach00
ee680ca890f8949f9ebd723c81eb789309a007301e6a621d431d367577

e =
0x257ae0a7b65e25b4e6de59c94259c86238666858419de8565c6bfc5ce2c7964bf013918b888341361ab61143a3848e34c8609b8eed9f44bbad6ec5c2e9c5d1fdc79038a1fcb
87aad9cae2cd885e2f42758b3ccc1efd37ebddc4462f86af68c199ae6015600077ddfb68e2dfa5ce5b9701aebbd2d1385171c300a2823d72511f5663692b0c33c7d5ccdc9acd25
5a0ef7d644d94670a7af40a6829494e8a8371657b8b96618b4323030713b32c29b159ebddd7a0af42c64c63b5d92a8b8ff493b0418828206fbb5663d35d7d42d03e54b37e08f
858296234d85d1eb72ad6fdff7115799c4901635cfd2ac83dda30b69fd9fbb9d1bddf38fd9c62e01776ab3c155d0e3fecefb7b9801e0cbc477b10dcd25f872c0f3172cdf05f27a1
7ea6d418f77bec591e1828b3830d35a3217cf59c7c384eee460579fff5d61f368bf453b142952456db7934d0063c0f46d9b5a8b35a7e7a4b211c532b7ce7ca5c09e45e241f87eaf
63e0ecd35ece6275509c999415138daa6530297564e32b298ddaf62d87bf

a =
0xd05e30b62b9c8a473b7513ac2c8b372d58092733a463458f749b2db4544a868344f79312dd03c366f6855f12db68881d3c57ba857c8c4927a93e8e59569ba1c8ead4720adc
c818b8fb92c3e0b23ab26704d7859c16ef09c9d2692f0132295fa8a67e7936d1a7d53872c3a51e7115cb952c3ccbbfeb8d29c2f5b58ecd799b423ebeca3725ff317682750610e
5986b9c0fb385feacd7bfa4082b8e820a7651e9569642d598c50050176019ade5ef446b5165b8280cf0c3f33efed0d13298aa6ab628669493d12bbbbb6e80bc220b303874f0b4a
cd8a61319215fb44decae0c2f116e0710d3b022baeb6ee294206310e42dd51c1de91de872f1adf5c395a674cf05

g =
0x23e774b678534aca7671d71a4f87dca2274e161b1f8561ab2ad51d12333d901f054759281a8bc1f9686184b5487e8353813dc0938e1570aed83f34857744fbc5ea856687dca
c06c53e3f9fc7e973cc94efb17ba85ecc00109e289b157172a0cad89ef4b6d795271e4e42aca26f0a0ee3cd71e2f561657fca7ddcb0fb45bb4d0491dca103cb5a4bfbfbf086bb58f
73ae4d0fbd5fd8b2baf3606aca6937f0cd4a764be6ab44a014c6be5d2614a9de6c6548ce86588b7a3cff8e326429c9c2dcdd8595d9c5703f91c07b835721d5b59d9ba26e3f4d34f
7e38a2d39602259cb024da307715f61a0c07ea4ce47eb19356109e4335820faa5b6e1f68fdcd4780c92a1c

c1 =
0x48eeea8b6d8ee7b7264115858c883005b93a515db96c33329f21253b5646d771ae09fd5e34338640ddd3bb933e1eda111625f527725d3e3b8a7990560a4a50271fce090fcd
1f525a9ddb5ceac20646afa5d383a84eab00c88fb6041899ebc1a6ce0b10a2fe00a7b6f7c0a23d6bb4c02237a4eaf7a8b8a748e1273a873d8b0b48526a6b96675163bd20d45205
0d530bd2cad6849a7087663a3aa3d11a2063716e4bd0279f1c369538d491b891301126310abb5b3b2ed1c2d4547fecdd0feba29fa1a14b641e2cb5669cc6225a0f9c573ef3dd2a
4b8f820afe63a639538d4dc52c1b76380ed3f6e9507c40f1a68412585acea0e5a8b5f00bfb15187ed84a65db1dba5e2c2a8234cc3acb15b354c3072e5f9e330f12aa6a323e4f981
85050a161621fcaa6a0067697420c5a9509ed35724a623e36963fa80a5831d6925b2a89be18935147cfc2d390cbe39f3e44a4951f3a6c078c10d2c241c6e9a1e84188b36abee3
cd456d1325ba73b12c54844a2deb193d2d5374679f9aaa668278893e0ef65c

c2 =
0xb9aa388ec95aa402d590ffd57ccb8cc1c741060cf4a33147ff5b030b8ab160d7475befb59a5221ab48a4567977f40ce5a60ec32ed35d21f05f3d9a3c4f0cc129f8bb71193cd14
752e9b446132fca31402fc38d32b41b5cd98b8d56b03e360ab778158f2b529b95adcef7014fb7914958f73eb866232d58c06034c9050e94dd47bca4b43b7d3c9990171b68fcd
26ec22779a229cb5d89b46110bfb17efcd68af786502a3e3e94dfac11b30108109ed331d2b3d69a70716d48420f0c1b43e954a09d1bf8a47f73360db789df6fdcf5ae360fc6d
2d5952e41f1ce4def3ce2454007711c7c0fd4359c30fea7357e6ec1c5e61a42e908c0d01cb145795b44bf505

cf_convergents = continued_fraction(e/n).convergents()
pt = 1337
ct = pow(pt, e, n)

for el in cf_convergents:
    k = Integer(el.numerator())
    d = Integer(el.denominator())

    if k == 0 or (e*d - 1) % k != 0:
        continue

    phi = (e*d - 1)//k
    d = inverse_mod(e, phi)
    if pow(ct, d, n) == pt:
        print(f"PHI: {phi}")
        break
else:
    print(f"[-] FAILURE")
    exit()

FF = GF(a)
k = pow(c1, d, n)
m = hex(int((c2 * inverse_mod(int(pow(g, k, a)), a)) % a))[2:]
print(f"root password: {binascii.unhexlify(m).decode()}")
```

Recover the flag

By executing the script the root password is printed to stdout :)

```
(sage-sh) Vittorio@DESKTOP-6CR3PAL:~$ sage /home/sage/Desktop/Writeups/nuclear_disaster/05_wiener.sage
PHI:
238416630167529620704407590936038900819846982570058698086517757645928724227378482546046640540040577228040064823545067461
930205107319856676309179617257491212344496124043669982524994484065315267964962736796485978040306408279072959657232931276
844588803643278547816327879367928419609120791444695797148057591301467279911750927584660375685580128341208149413365742677
323929015758088387758497817388155078254793692352092943776372870066922564908349792147204162179607548897820067526503837689
963252002329048211383220541315031200968178018189049918491833272320612359646503610964074400809919146931519417201182898431
794094926794962970966193015053856387456498960461940331576084636791544078178686131441380730764383886228478775233365520834
579975896286117332559405116879395779877380603708854342624746132297859395869335922215360995551425614899234306866424782363
6057162567376161154249363899840728374219789666190447914045676061514338490901326171640
root password: 3xtr4H34t1nr34ct0r
```

as root we can read the flag:

```
root@2da296184d7b:~# ls -alh
total 24K
drwx----- 1 root root 4.0K Nov 22 20:41 .
drwxr-xr-x 1 root root 4.0K Nov 22 19:32 ..
-rw-r--r-- 1 root root 3.1K Dec  5 2019 .bashrc
drwx----- 2 root root 4.0K Nov 22 20:41 .cache
-rw-r--r-- 1 root root 161 Dec  5 2019 .profile
-rw-r--r-- 1 root root 40 Nov  9 13:06 flag.txt
root@2da296184d7b:~# cat flag.txt;echo
HTB{s3cur3_y0ur_cr1t1c4l_1nfr4s7ruc7ur3}
```

CHEESE!

```
root@2da296184d7b:~# ls -alh
total 24K
drwx----- 1 root root 4.0K Nov 22 20:41 .
drwxr-xr-x 1 root root 4.0K Nov 22 19:32 ..
-rw-r--r-- 1 root root 3.1K Dec  5 2019 .bashrc
drwx----- 2 root root 4.0K Nov 22 20:41 .cache
-rw-r--r-- 1 root root 161 Dec  5 2019 .profile
-rw-r--r-- 1 root root 40 Nov  9 13:06 flag.txt
root@2da296184d7b:~# cat flag.txt;echo
HTB{s3cur3_y0ur_cr1t1c4l_1nfr4s7ruc7ur3}
root@2da296184d7b:~#
```