Writeup for Wargames.my 2021

Writeup for Wargames.my 2021

Category: Cryptography - easyrsa

Given python file

chal.py

In the source code value for n, c and hint is given,

`hint = p*q-p-q+1`

If we factor out `p*q-p-q+1` we will get `(p-1)*(q-1)` which is `PHI`. Therefore `PHI=hint`.

```
from Crypto.Util.number import long_to_bytes
import sys

c=326595170717224270972747273938687349470324991228550526537114639319603037
241378180393016466314937222843933373386508233001164270491309124965349720003
780944335044675673331902347399251034625522516076080289558400382270145968250
404840010571742127988310589087523302675740007385714132298726983039469798322
204553974625219330303100743678921966397869657279014124458772194638540183755
555648059230249319084653095611900234769213541458191562314660299246403058564
728320063378773616082325606774278596866388424931701492973081462101902738956
895920426343367096059611851871333644229929157413629018144692136808944126
```

```
707871537758673177856001 07
n=1830431349962727887249734710678108876584497175292449493658113729439925159812205449197035262499780489159736857215197519901287536189286237883428568365290300704494705561934268483004388738852019888998689790108244740590340640759487921847748601030394915058406201374300210222302769593330557847427940254351576533641220800685905439353457084120314402103414020774365975645635517687365866943105081794435436954394318007100447898862116757879430973808817543741012812323319251355833960761042309275881707711678237040581426350521574948267485288925201758969653042254739312397862488388906972755758521087930531988330137181620552809043440 7
#PHI =(p-1)*(q-1) = hint = p*q-p-q+1
PHI=1830431349962727887249734710678108876584497175292449493658113729439925159812205449197035262499780489159736857215197519901287536189286237883428568365290300704494705561934268483004388738852019888998689790108244740590340640740759487921847748601030394915058406201374300210222302769593330557847427940254351576533641193718829085241920609043746758562546428726601779051836945819727406203022234361932871768234345227095844165965086543100805475233734788611736563726752110713668306358343947215223729066594029304583431226741472301578327786047842152745501205045067602882077634449040311631828852598470552682455871506975125259548912060 0
e=65537
d=pow(e, -1, PHI) #inverse mod
res=pow(c,d,n)

print ("Cipher: ",c)

print ("\n=== Calc ===")
print ("d=",d)
print ("n=",n)
print ("Decrypt: %s" % ((long_to_bytes(res))))
```

```
5645635517618605866943105081794435436954394318007100447892758817077116782370405814263505215749482674852889252017718162055280904344078
Decrypt: b'wgmy{227d1562df0d940d94d75b0512f4bc6c}'
PS D:\CTF\Wargames2021>
```

Category: Cryptography - hohoho

Given server to connect and its soure code

server.py

In the menu it contains Register, Login, Make a wish, Wishlist (Santa only) and Exit.

In register you can register any string as long as the string did not contains any `Santa` in the string. Once register you will receive token which is md5 hash of `SECRET+stringofyourname`.

In login, you can login with your name and token given to login. From the source code, there is no database that keep track wether you register or not. Since it only, validate the login based on md5 hash of `SECRET+stringofyourname` which mean that you can login as any string if you know the md5 hash of it but since we do not know the `SECRET`, thus we cannot do that unless with an attack which we will explain later.

Make a wish menu can be ignore, and last is Wishlist menu will open file call `wishes.txt` that contains the flag but only if your user name contains `Santa`.

Based on the way that md5 hash is setup it is vulnerable to Hash Length Extension Attack.

`Hash = md5(unknownsecret+data)`

When a Merkle–Damgård based hash is misused as a message authentication code with construction `H(secret ‖ message)`, and message and the length of secret is known, a length extension attack allows anyone to include extra information at the end of the message and produce a valid hash without knowing the secret.

`ValidHash = md5(unknownsecret+data+padding+appenddata)`

By using HashPump

```
from pwn import *
import hashpumpy

secretlen = 8 #key_length

r = remote('13.213.3.148', 1337)

t = r.recvuntil(b"option: ")
r.sendline(b"1")
t = r.recvuntil(b"name: ")
r.sendline(b"kaitorque") #original_data
t = r.recvuntil(b"login: ")
t = r.recvline().decode().strip()
token = t #hexdigest
print("token", token)
# hashpumpy.hashpump(hexdigest, original_data, data_to_add, key_length)
# return -> (digest, message) tuple
hpump = hashpumpy.hashpump(token, 'kaitorque', 'Santa', secretlen)

print(hpump[0]) #valid hash
```

```
print(hpump[1]) #data+padding+appenddata

t = r.recvuntil(b"option: ")
r.sendline(b"2")
t = r.recvuntil(b"name: ")
r.sendline(hpump[1]) #message
t = r.recvuntil(b"token: ")
r.sendline(hpump[0]) #digest

t = r.recvuntil(b"Exit\n")
print(t)
r.sendline(b"4")
t = r.recvuntil(b"}")
print(t)
```

```
[+] Opening connection to 13.213.3.148 on port 1337: Done
token c3e0b0e9150325c9204dcfb34547c090
fec394dc5b8deb670e6eb017b5fe64bc
b'kaitorque\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x88\x00\x00\x00\x00\x00\x00\x00Santa'
serverho2.py:27: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#by
tes
  r.sendline(hpump[0])
b'Login successfully as kaitorque\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x88\x00\x00\x00\x00\x00\x00\x00Santa\n
Now you can make a wish!\n\n1. Register\n2. Login\n3. Make a wish\n4. Wishlist (Santa Only)\n5. Exit\n'
b'Enter option: Wishes:\nSanta Claus: Merry Christmas! Flag: wgmy{6bd7f862cbfa8b802a63b09979d00ee6}'
[*] Closed connection to 13.213.3.148 port 1337
```

Category: Web - Mountain

Given a file and its link to the website

generate.php.bak

In the file contains genpassverify function where verify code is randomly selected from `mt_rand(1000000000,9999999999);` and which later will be use for seed for `mt_rand()` using `mt_srand($verify_code);` where the account password is generate from.

In the web contains register and login, when you register you will get return message said

```
Success! Hello Kaitorque, your password is: 1551655900
You need to activate your account first. Check your email for
activation/verification link.

The link might look something like:
https://mountain.wargames.my/verify.php?username=henson&verify=1929258756
```

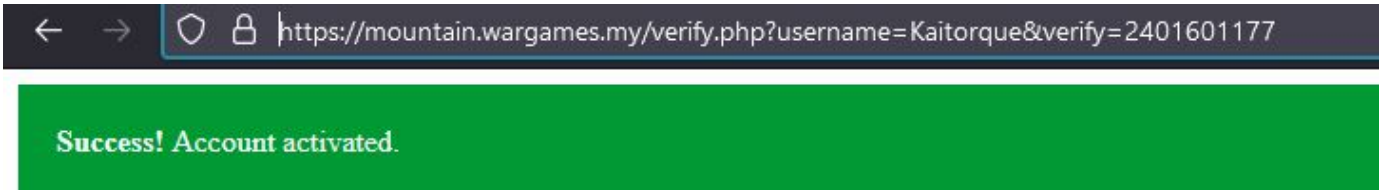
Unfortunately you need to be activate/verify to login.

Therefore from the code, we can get the verify code if we know the seeds.

By using php_mt_seed we can bruteforce to find the seed.

```
kaitorque-kali@kaitorque-kali:~/Desktop/Wargames2021$ ./output 1551655900
Pattern: EXACT
Version: 3.0.7 to 5.2.0
Found 0, trying 0x90000000 - 0x93ffffff, speed 165.8 Mseeds/s
seed = 0x921ef12a = 2451501354 (PHP 3.0.7 to 5.2.0)
seed = 0x921ef12b = 2451501355 (PHP 3.0.7 to 5.2.0)
Found 2, trying 0xfc000000 - 0xffffffff, speed 165.5 Mseeds/s
Version: 5.2.1+
Found 2, trying 0x08000000 - 0x09ffffff, speed 1.3 Mseeds/s
seed = 0x0875e51e = 141944094 (PHP 5.2.1 to 7.0.x; HHVM)
Found 3, trying 0x0e000000 - 0x0fffffff, speed 1.3 Mseeds/s
seed = 0x0f183cc8 = 253246664 (PHP 7.1.0+)
Found 4, trying 0x8e000000 - 0x8fffffff, speed 1.3 Mseeds/s
seed = 0x8f258699 = 2401601177 (PHP 7.1.0+)
Found 5, trying 0x98000000 - 0x99ffffff, speed 1.3 Mseeds/s
```

Put it to the url we got our user to be verify and activated



Then we can use the login with the username and password given to get the flag