# VishwaCTF

CHALLENGE NAME : [YOUR BONUS]

DEV : [ADITYA GAIKWAD ]

CATEGORY : [REVERSE ENGINEERING ]

LEVEL : [EASY-MEDIUM]



**2024**

Description: I am very kind and your friend also. I was going to give you some flags; But there was a ransomware attack on that flag-containing file, And all those flags Encrypted by that ransomware. after cross checking that directories I got ransomware file and some other stuffs. I'm gonna to be share that stuffs with you. Because of that ransomware I'm unable to give you flags 🥺 🥺 ; now I want recover that flags, can you help me ... please 🥺 🥺 ... Your assistance would be greatly appreciated, and you will receive a reward for your help.

Let's analyze this ransomware...

```
61  local_6d = '9';
62  puVar2 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_6d);
63  *puVar2 = 0x26;
64  std::__cxx11::basic_string<>::basic_string();
65  local_14 = 0;
66  std::basic_fstream<>::open((basic_string *)&asd[abi:cxx11],0x10);
67  while( true ) {
68    piVar3 = (int *)std::getline<>(local_1a0,local_1b8);
69    bVar1 = std::basic_ios::operator.cast.to.bool
70                      ((basic_ios *)((int)piVar3 + *(int *)(*piVar3 + -0xc)));
71    if (!bVar1) break;
72    std::__cxx11::basic_string<>::basic_string(local_1b8);
73    std::__cxx11::basic_string<>::basic_string(local_a0);
74    local_2fc = &stack0xfffffe28;
75    devil_function((basic_string)local_6c,(basic_string *)local_2fc);
76    std::__cxx11::basic_string<>::~basic_string(local_6c);
77    std::__cxx11::basic_string<>::length();
78    local_2fc = &stack0xfffffe28;
79    zarathos((basic_string *)&stack0xfffffe10,(int)local_2fc);
80    std::__cxx11::basic_string<>::basic_string((basic_string *)&stack0xfffffe28);
81    local_2fc = (char *)local_14;
82    local_24 = Lucifer((basic_string)local_54,local_14);
83    std::__cxx11::basic_string<>::~basic_string(local_54);
84    ghost_ridders_wepon();
85    local_2fc = (char *)local_14;
86    matter_manipulation[abi:cxx11]((int)&stack0xfffffdf8);
87    std::__cxx11::basic_string<>::basic_string((basic_string *)&stack0xfffffdf8);
88    Trigon((basic_string)local_3c);
89    std::__cxx11::basic_string<>::~basic_string(local_3c);
90    local_14 = local_14 + 1;
91    std::__cxx11::basic_string<>::~basic_string((basic_string<> *)&stack0xfffffdf8);
92    std::__cxx11::basic_string<>::~basic_string((basic_string<> *)&stack0xfffffe10);
93    std::__cxx11::basic_string<>::~basic_string((basic_string<> *)&stack0xfffffe28);
94  }
95  std::basic_fstream<>::close();
    std::basic_fstream<>::close();
```

Our data encrypted by some function which called in main functions while loop.

Queue of function call in this while loop is like devil_function--zarathos—Lucifer--ghost_ridder_wepon—matter_manipulation—Trigon...

After checking devil_function it seems like nothing... So move to next func.

In zarathos func. After analyzing that we can see that it used for just rotating strings by random rotations and start=3 and end=length-1, random rotation in -3 to 3.

```
41  puVar3 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_29);
42  *puVar3 = 0x28;
43  local_28 = std::__cxx11::basic_string<>::begin();
44  _Var1 = (__normal_iterator)
45          __gnu_cxx::__normal_iterator<>::operator+((__normal_iterator<> *)&local_28,local_1c);
46  _Var2 = (__normal_iterator)std::__cxx11::basic_string<>::begin();
47  std::reverse<>(_Var2,_Var1);
48  _Var1 = (__normal_iterator)std::__cxx11::basic_string<>::end();
49  local_24 = std::__cxx11::basic_string<>::begin();
50  _Var2 = (__normal_iterator)
51          __gnu_cxx::__normal_iterator<>::operator+((__normal_iterator<> *)&local_24,local_1c);
52  std::reverse<>(_Var2,_Var1);
53  _Var1 = (__normal_iterator)std::__cxx11::basic_string<>::end();
54  _Var2 = (__normal_iterator)std::__cxx11::basic_string<>::begin();
55  std::reverse<>(_Var2,_Var1);
56  local_1d = '2';
```

So, Now move on next one ... Lucifer(string , int)

```
local_1f = '7';
puVar2 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_1f);
*puVar2 = 0x21;
local_38[0] = -3;
local_38[1] = 0xfffffffe;
local_38[2] = 0xffffffff;
local_38[3] = 1;
local_28 = 2;
local_24 = 3;
local_10 = random_pick(4,0);
local_3c = local_38[local_10];
local_1e = '3';
puVar2 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_1e);
*puVar2 = 0x40;
local_14 = param_1;
local_40 = std::__cxx11::basic_string<>::begin();
local_44 = std::__cxx11::basic_string<>::end();
while( true ) {
  bVar1 = __gnu_cxx::operator!=((__normal_iterator *)&local_40,(__normal_iterator *)&local_44);
  if (!bVar1) break;
  uVar3 = __gnu_cxx::__normal_iterator<>::operator*((__normal_iterator<> *)&local_40);
  local_15 = *(char *)uVar3;
  local_1c = local_15 + local_3c;
  adfedd(extraout_ECX,(int)((ulonglong)uVar3 >> 0x20));
  __gnu_cxx::__normal_iterator<>::operator++((__normal_iterator<> *)&local_40);
}
local_1d = '8';
puVar2 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_1d);
*puVar2 = 0x25;
return 0;
```

You can see that there is an int array contains {-3,-2,-1,1,2,3} . and random_pick(4,0) function is used for getting random value from that array and

Initialized new int variable local_15 with ascii value of each char of string argument and that randomly generated value is added to it, after that adfedd(int , int) function is called …

```
3
4  void __fastcall adfedd(int param_1,int param_2)
5
6  {
7    basic_string local_24 [7];
8
9    std::__cxx11::to_string((__cxx11 *)param_1,(int)local_24);
10   std::__cxx11::basic_string<>::append(local_24);
11   std::__cxx11::basic_string<>::~basic_string((basic_string<> *)local_24);
12   return;
13 }
14
```

In it one argument is converted to string local_24 and that local_24 was append to other unknown string … it may be that string which containing all ascii values

Now moving to next function call … ghost_ridders_wepon().

```
5  void ghost_ridders_wepon(void)
6
7  {
8    undefined *puVar1;
9    char local_e;
10   char local_d [9];
11
12   local_e = '4';
13   puVar1 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_e);
14   *puVar1 = 0x5e;
15   local_d[0] = '0';
16   puVar1 = (undefined *)std::map<>::operator[]((map<> *)&_HM,local_d);
17   *puVar1 = 0x2a;
18   return;
19 }
20
```

Nothing special there …just adding value to map so let’s trace that whole map.

After going through whole .exe I found some parts of map…

There too many like this one so note it down ...

```
Open
1   local_e = '4';
2   puVar1 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_e);
3   *puVar1 = 0x5e;
4
5   local_d[0] = '0';
6   puVar1 = (undefined *)std::map<>::operator[]((map<> *)&_HM,local_d);
7   *puVar1 = 0x2a;
8
9   local_1f = '7';
0   puVar2 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_1f);
1   *puVar2 = 0x21;
2
3   local_1e = '3';
4   puVar2 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_1e);
5   *puVar2 = 0x40;
6
7   local_1d = '8';
8   puVar2 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_1d);
9   *puVar2 = 0x25;
0
1   local_2b = '5';
2   puVar3 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_2b);
3   *puVar3 = 0x23;
4
5   local_2a = '1';
6   puVar3 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_2a);
7   *puVar3 = 0x29;
8
9   local_29 = '6';
0   puVar3 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_29);
1   *puVar3 = 0x28;
2
3   local_1d = '2';
4   puVar3 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_1d);
5   *puVar3 = 0x24;
6
7   local_6d = '9';
8   puVar2 = (undefined *)std::map<>::operator[]((map<> *)&_HM,&local_6d);
9   *puVar2 = 0x26;
0
1  0 1 2 3 4 5 6 7 8
```

Finally ...

```
'0' = *
'1' = )
'2' = $
'3' = @
'4' = ^
'5' = #
'6' = (
'7' = !
'8' = %
'9' = &
```

Let's move on next function matter_manipulation()

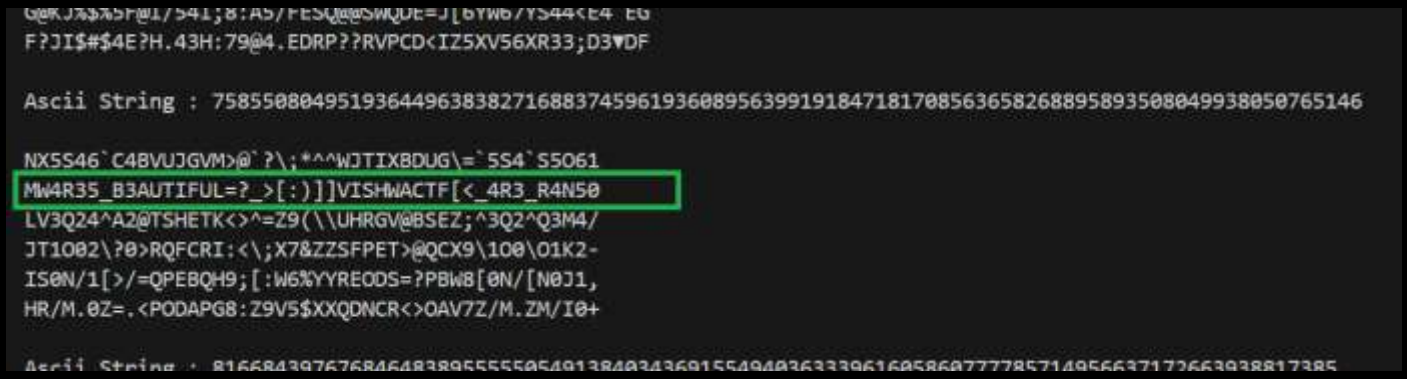
```
12  char local_25;
13  vector<> local_24 [14];
14  char local_16;
15  char local_15;
16  char local_14;
17  char local_13;
18  char local_12;
19  char local_11;
20  undefined1 *local_10;
21
22  std::__cxx11::basic_string<>::basic_string();
23  std::vector<>::vector(local_24);
24  local_16 = 'V';
25  std::vector<>::push_back(local_24,&local_16);
26  local_15 = 'i';
27  std::vector<>::push_back(local_24,&local_15);
28  local_10 = &sss[abi:cxx11] + in_stack_00000008 * 0x18;
29  local_2c = std::__cxx11::basic_string<>::begin();
30  local_30 = std::__cxx11::basic_string<>::end();
31  while( true ) {
32    bVar1 = __gnu_cxx::operator!=((__normal_iterator *)&local_2c,(__normal_iterator *)&local_30);
33    if (!bVar1) break;
34    pcVar2 = (char *)__gnu_cxx::__normal_iterator<>::operator*((__normal_iterator<> *)&local_2c);
35    local_25 = *pcVar2;
36    local_14 = 'h';
37    std::vector<>::push_back(local_24,&local_14);
38    local_13 = 'a';
39    std::vector<>::push_back(local_24,&local_13);
40    pcVar2 = (char *)std::map<>::at((map<> *)&_HM,&local_25);
41    std::__cxx11::basic_string<>::operator+=((basic_string<> *)param_1,*pcVar2);
42    __gnu_cxx::__normal_iterator<>::operator++((__normal_iterator<> *)&local_2c);
43  }
44  local_12 = 'w';
45  std::vector<>::push_back(local_24,&local_12);
46  local_11 = 's';
47  std::vector<>::push_back(local_24,&local_11);
48  std::vector<>::~vector(local_24);
49  return param_1;
}
```

There is only one important thing is, That ascii value string is just mapped with our map. And ransomware did his job with our data,

And after that it just write that hacked_data in new file and deleted the original file….

So by walking over the above analysis I written a java code, and I'm attaching that file

After running that programme on your_hacked_data I got flag string…




String which containing flag is :-
MW4R35_B3AUTIFUL=?_>[:)]]]VISHWACTF[<_4R3_R4N50

So, order it in flag formate VISHWACTF[]

VISHWACTF[<_4R3_R4N50MW4R35_B3AUTIFUL=?_>[:)]]

So, after that; As per an instruction given at time of ctf final flag is

VISHWACTF[4R3_R4N50MW4R35_B3AUTIFUL].