pwnthem0le is a Turin-based, hacking students group born out of CyberChallenge 2018. Read more about us and how to join us!

18 October 2018

PicoCTF 2018 - be-quick-or-be-dead WriteUp

by XxcoralloxX

be-quick-or-be-dead are 3 similar reverse challenge.

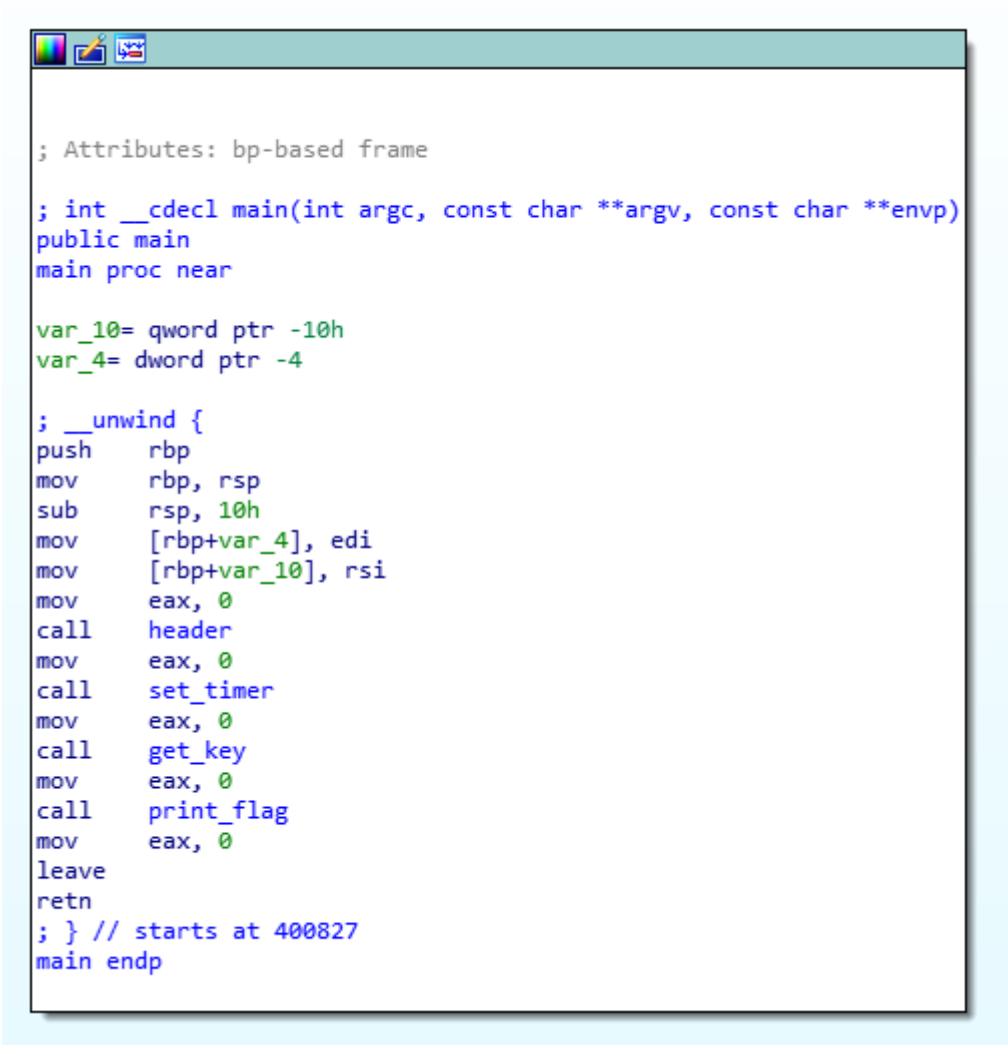
be-quick-or-be-dead-1

Running the program we see a key “being calculated” After a few seconds of execution, a message says that we need a faster machine and end the process.

```
root@hlkali:/home/hacker/Desktop/CTF/PicoCTF2018# ./BeQuick1
Be Quick Or Be Dead 1
=====================

Calculating key...
You need a faster machine. Bye bye.
root@hlkali:/home/hacker/Desktop/CTF/PicoCTF2018#
```

It’s quite simple to see with ida what’s happening



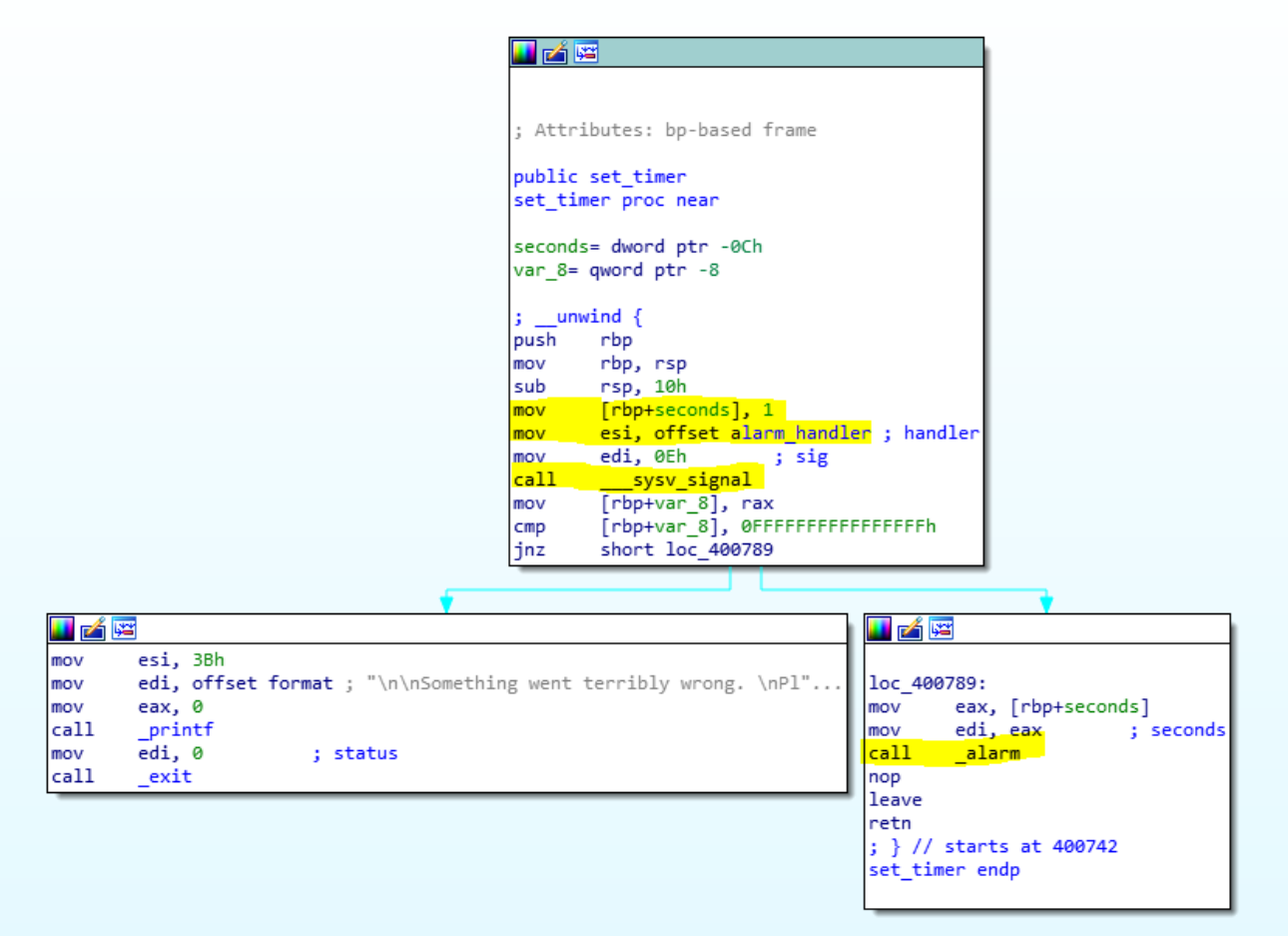
```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_10= qword ptr -10h
var_4= dword ptr -4

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 10h
mov     [rbp+var_4], edi
mov     [rbp+var_10], rsi
mov     eax, 0
call    header
mov     eax, 0
call    set_timer
mov     eax, 0
call    get_key
mov     eax, 0
call    print_flag
mov     eax, 0
leave
retn
; } // starts at 400827
main endp
```

A timer is set, after some time (1 second) it will send a signal to the program to stop the execution, and alarm_handler will be execute



Okay, the first thing we can do is try to give more time before the signal. so, we just patch the “mov [rbp+seconds], 1” into “mov [rbp+seconds], 8” and see what’s happen

Here we are after a few seconds we get the flag!

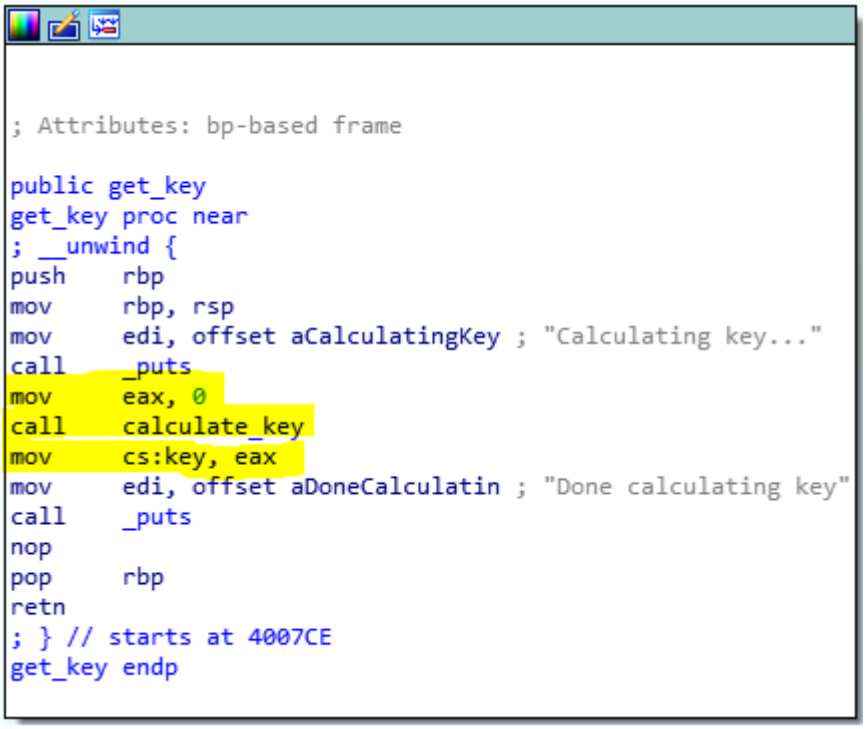
```
root@hlkali:/home/hacker/Desktop/CTF/PicoCTF2018# ./BeQuick1_patched
Be Quick Or Be Dead 1
=====================

Calculating key...
Done calculating key
Printing flag:
picoCTF{why_bother_doing_unnecessary_computation_402ca[redacted]5}
root@hlkali:/home/hacker/Desktop/CTF/PicoCTF2018#
```

be-quick-or-be-dead-2

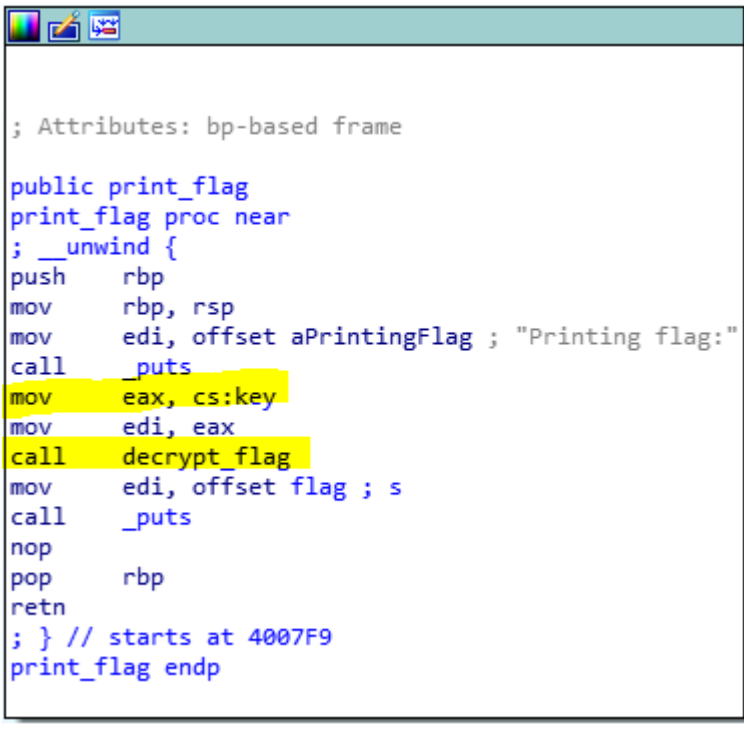
This is very similar, we have the same timer, this time giving us 3 seconds. But the patch applied before isn’t working, (i let it run for more than 5 minutes). In this case, we need to look deeper.

In particular, we should take a look at what makes the program so time-demanding. The main calls a function “get_key” which calls a function “calculate_key”. The return value of calculate_key is moved into eax and will be used later, from the function “decript_key” to decrypt it. So we can’t just skip that.



```
; Attributes: bp-based frame

public get_key
get_key proc near
; __unwind {
push    rbp
mov     rbp, rsp
mov     edi, offset aCalculatingKey ; "Calculating key..."
call    _puts
mov     eax, 0
call    calculate_key
mov     cs:key, eax
mov     edi, offset aDoneCalculatin ; "Done calculating key"
call    _puts
nop
pop     rbp
retn
; } // starts at 4007CE
get_key endp
```

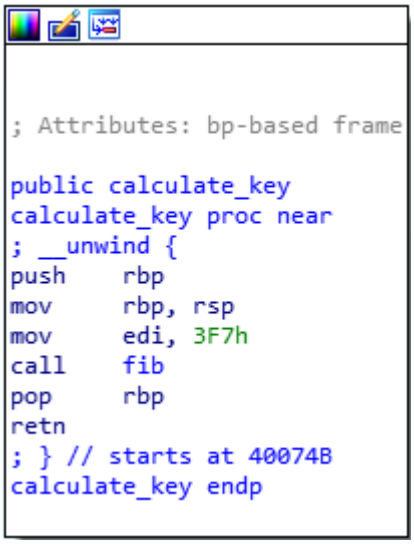


```
; Attributes: bp-based frame

public print_flag
print_flag proc near
; __unwind {
push    rbp
mov     rbp, rsp
mov     edi, offset aPrintingFlag ; "Printing flag:"
call    _puts
mov     eax, cs:key
mov     edi, eax
call    decrypt_flag
mov     edi, offset flag ; s
call    _puts
nop
pop     rbp
retn
; } // starts at 4007F9
print_flag endp
```

Very good.

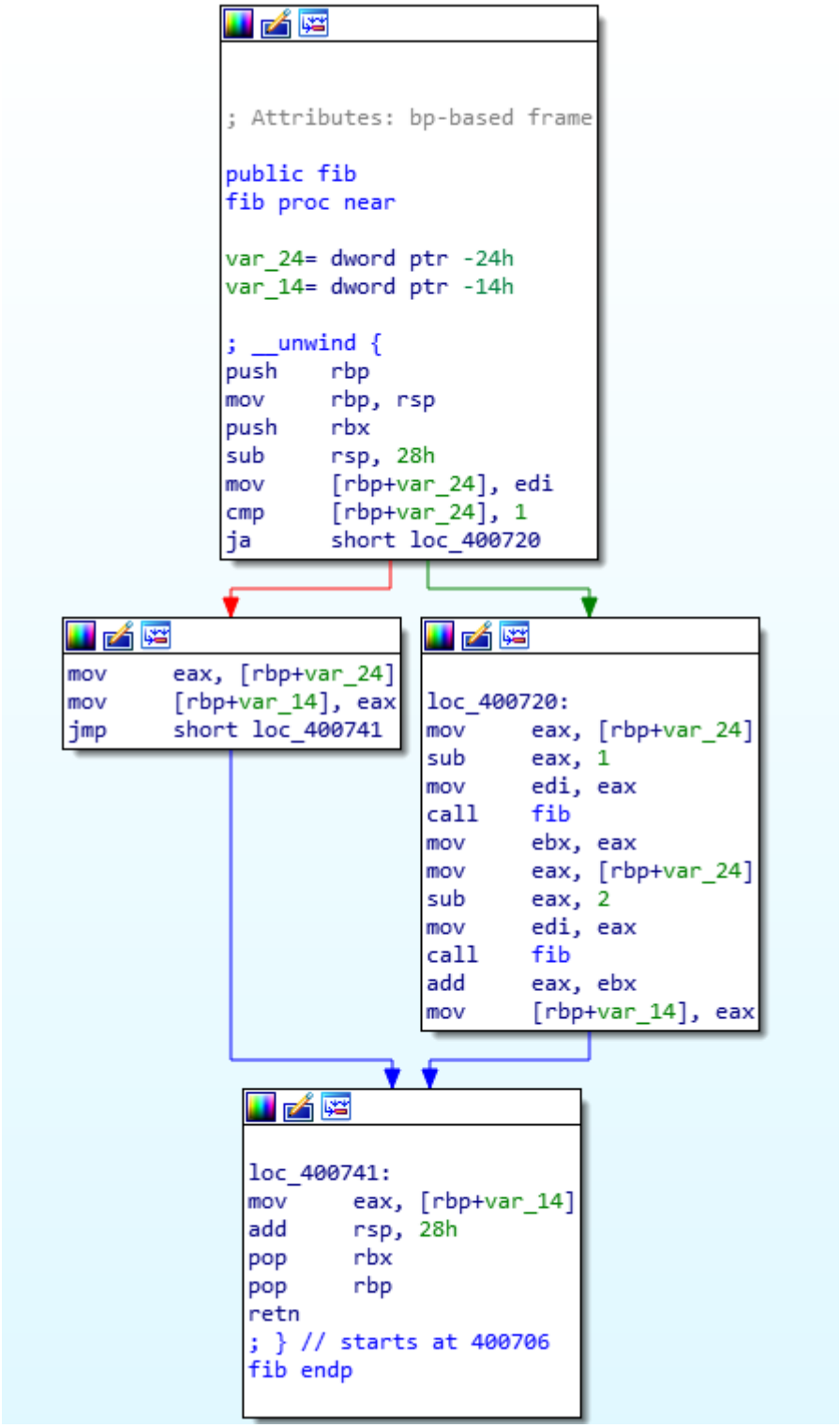
So, how this key is generated? Well, calculate_key it’s a one-instruction function it calls fib(3f7h) = fib(1015)



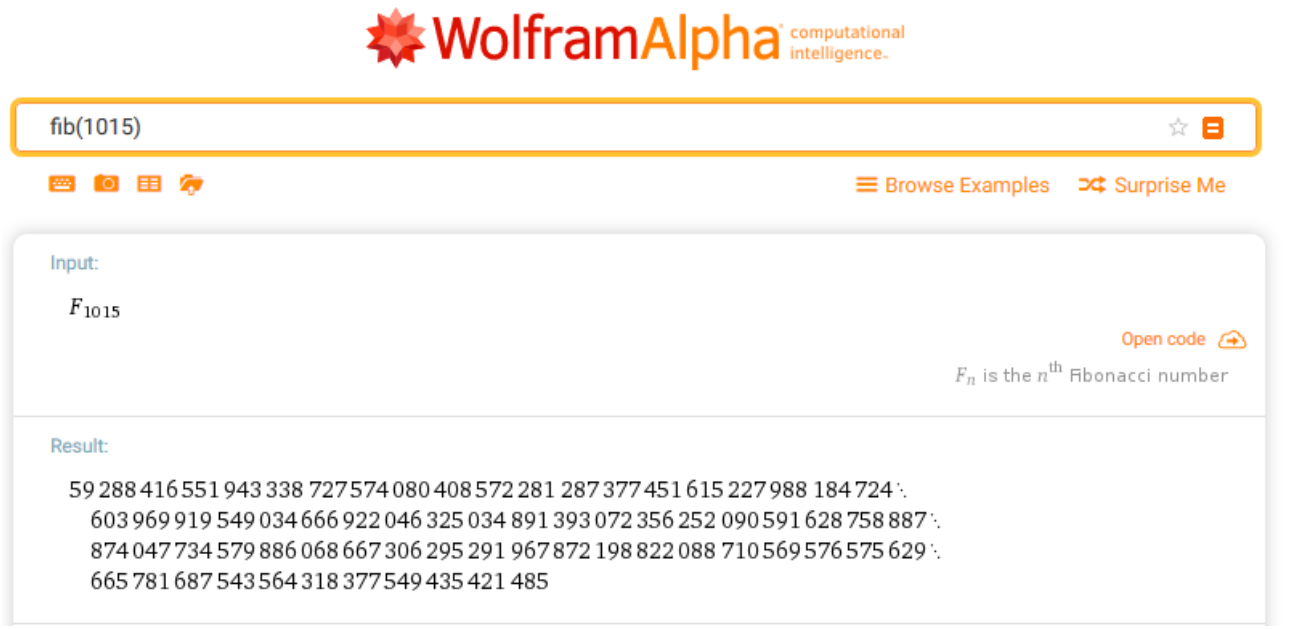
```
; Attributes: bp-based frame

public calculate_key
calculate_key proc near
; __unwind {
push    rbp
mov     rbp, rsp
mov     edi, 3F7h
call    fib
pop     rbp
retn
; } // starts at 40074B
calculate_key endp
```

The name should already raise suspicion, but looking at the implementation, it’s clear that it is a recursive version of Fibonacci function.



To calculate fib(1015) this isn't the right way since that implementation has an exponential computational complexity. We could write our function to do it better, or we could ask to WolframAlpha.




```
fib(1015) = 592884165519433387275740804085722812873774516152279881
```

At this point, the result would be stored into eax which is a 32-bit register, an int (in my architecture at least) is 32 bit too, so in order to quickly have the right number and ignore the overflow, I just put that result in an int variable, and make it print.

now we just want to patch the

“call fib” with “mov EAX, 3611214637”

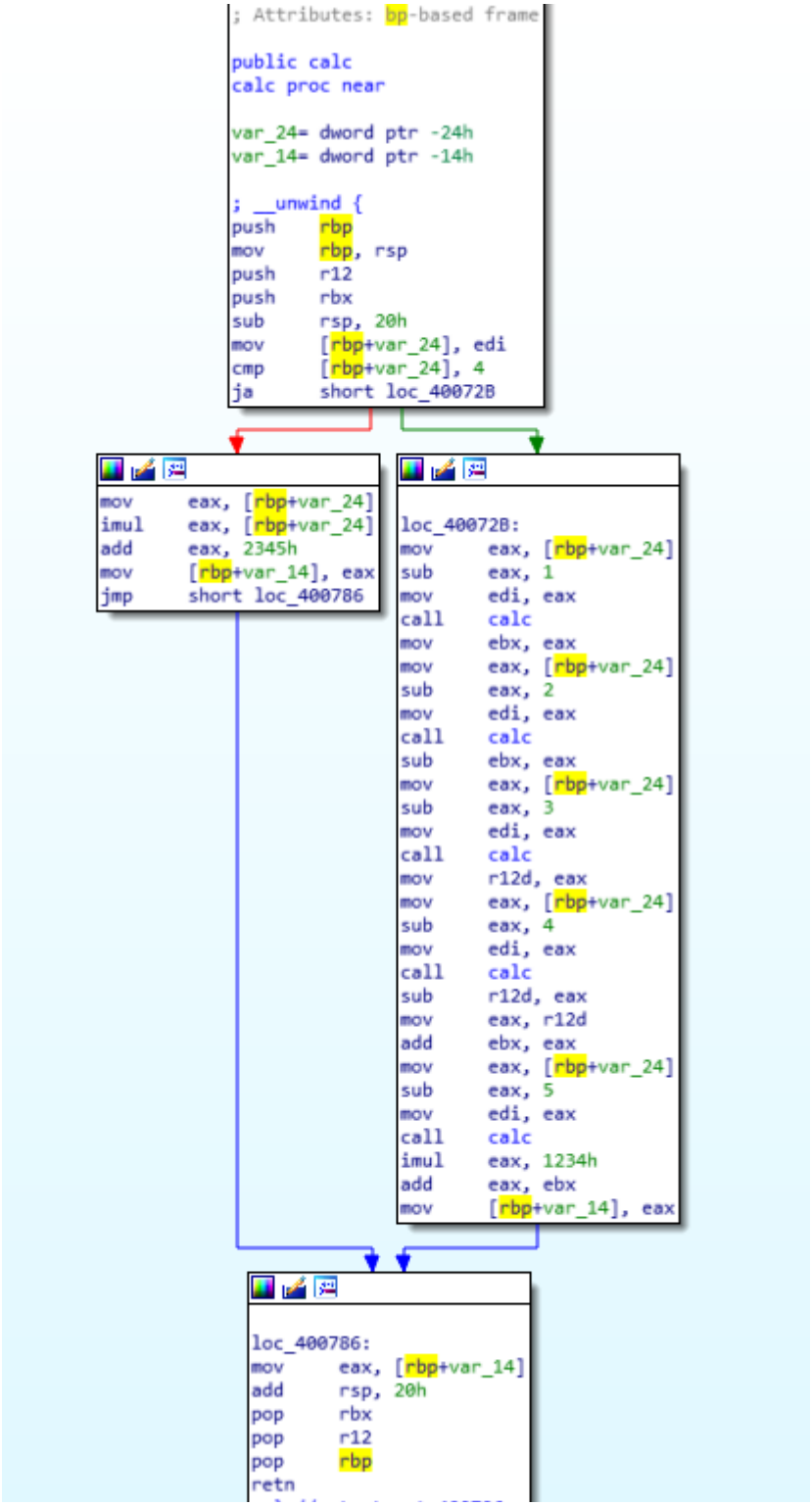
and we win.

```
root@hlkali:/home/hacker/Desktop/CTF/PicoCTF2018# ./be-quick-or-be-dead-2
Be Quick Or Be Dead 2
=====================

Calculating key...
Done calculating key
Printing flag:
picoCTF{the_fibonacci_sequence_can_be_done_fast_73e[redacted]e}
root@hlkali:/home/hacker/Desktop/CTF/PicoCTF2018#
```

be-quick-or-be-dead-3

Here we have again the same challenge, whit again a 3s timer, and a calculate_key too time-demanding. But this time, the function which calculates the key isn’t fib(), but a generic calc(). We need to reverse it.



Luckily it is quite simple:

in short, it does:

```
unsigned int calc(unsigned int x)
{
    unsigned int v1;
    unsigned int v2;
    unsigned int v3;
    unsigned int v4;
```

```
    unsigned int v5;

    if ( a1 > 4 )
    {
        v1 = calc(x - 1);
        v2 = v1 - calc(x - 2);
        v3 = calc(x - 3);
        v4 = v3 - calc(x - 4) + v2;
        v5 = v4 + 4660 * calc(x - 5);
    }
    else
    {
        v5 = x * x + 9029;
    }
    return v5;
}
```

okay, again an exponential-recursive funciton, which has as input 19965h.

no way it can’t be solved in this way.

We just need to rewrite it in a linear-complexity. (dynamic programming if you want):

This was my implementation:

```
#define N 104806
unsigned int v[N]={0};
int main() {

    int i;
    unsigned int v1,v2,v3,v4;
    v[0]=9029;
    v[1]=9030;
    v[2]=9033;
    v[3]=9038;
    v[4]=9045;
    for(i=5;i<N;i++){
        v1=v[i-1];
        v2=v1-v[i-2];
        v3=v[i-3];
        v4=v3-v[i-4]+v2;
```

```
        v[i]=v4+4660*v[i-5];
    }
    printf("%u",v[N-1]);
    return 0;
```

we take the result, keep only the lowers 16 bits, patch the program with: “MOV EAX,9E22C98Eh” as in be-quick-or-be-dead-2, and run it.

```
root@hlkali:/home/hacker/Desktop/CTF/PicoCTF2018# ./be-quick-or-be-dead-3
Be Quick Or Be Dead 3
=====================

Calculating key...
Done calculating key
Printing flag:
picoCTF{dynamic_pr0gramming_ftw_b5c45645}
root@hlkali:/home/hacker/Desktop/CTF/PicoCTF2018#
```

and that’s it. We solved all those 3 challenges, they were simple and nice!

XxcoralloxX

tags: reverse PicoCTF2018