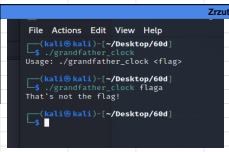
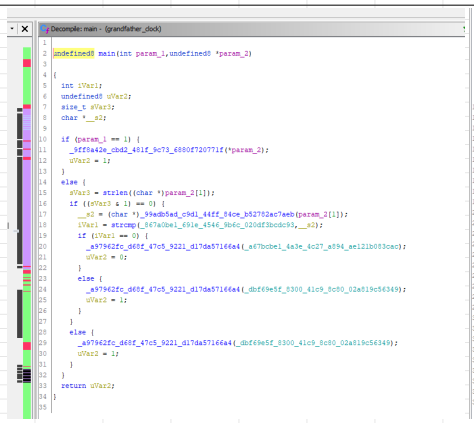
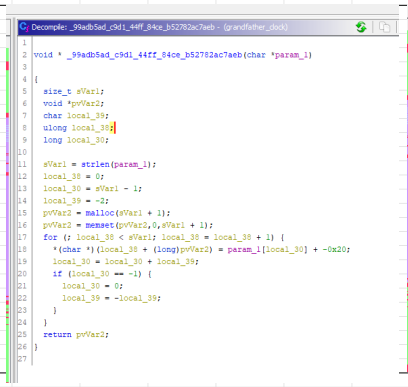
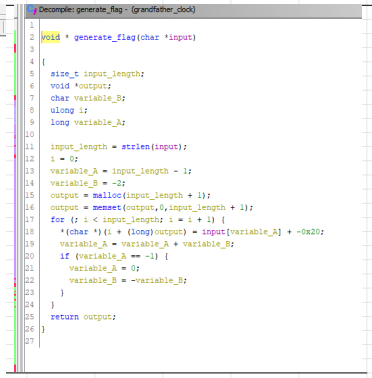
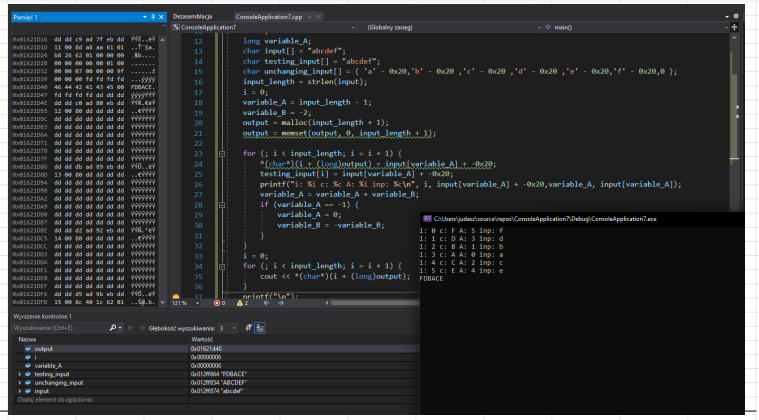
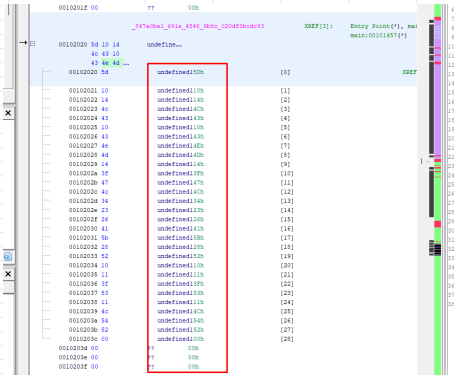
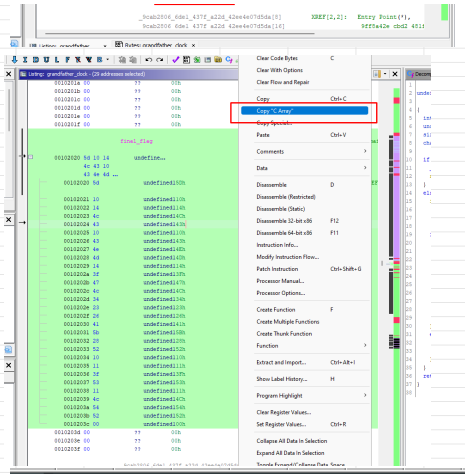
Autor:	Iodsb's
Tytuł:	Grandfather Clock
Język:	C/C++
Platforma:	Unix/linux
Trudność:	2.7
Jakość:	5.0
Arch:	x86-64
Link:	https://crackmes.one/crackme/60db74bb33c5d410b88430dc
Opis:	Just a classic reverse engineering CTF, grab the executable, figure out how it works,
Plik:	https://crackmes.one/static/crackme/60db74bb33c5d410b88430dc.zip

Krok	Rozwiązanie	Zrzuty Ekranu
1	Sprawdzenie jak działa plik .exe Działanie polega na flagi jako argumenty. Zwraca czy wartość jest poprawna czy nie.	
	Sprawdzamy funkcję main. Po prawej stronie pozmieniałem pare typów zmiennych. Możemy zauważyć parę rzeczy: 1) Nasz input musi być parzystej długości 2) Na bazie naszego **input** jest generowany **output** za pomocą funkcji **_99adb5....** 3) potem sprawdzamy czy są sobie równe za pomocą **strcmp** 4) jeżeli są to widzimy ze main zwraca **0**, więc tam jest najwiekse prawdopodobienstwo prawidłowej odpowiedzi Musimy sprawdzić co funkcja **_99adb5....** robi.	 
	Funkcja **_99adb5....** na bazie naszego **inputu** generuje **output** zmienna **output** rezerwuje sobie kawałek pamięci i ustawia go na **0** Czas przeanalizować jak działa ta funkcja	 
	Wykonałem adaptacje tej funkcji w **C++** i dla przykładowego **inputu = "abcdef"** otrzymamy **output = "FDBACE"** na początku widzimy że **-0x20** zamieni nam wejscie **abcdef -> ABCDEF** a potem w pętli zostanie ona poprzestawiana **ABCDEF -> FDBACE** dodatkowo wiemy ze input bedzie mial parzystą długość z funkcji **main** po konsoli i rejestrach widzimy ze iteracja wygląda w następujący sposób idziemy od *'fyf'* naszego napisu co drugi znak gdy dojdziemy do początku to wtedy idziemy w drugą stronę Na tym etapie juz wiemy dokładnie jak działa ta funkcja	
	Analizując maina dalej widzimy że zmienna **_s2** jest porównywana z jakimś segmentem pamięci	 

W celu przekopiowania gdzieś wartości tej zmiennej
warto skorzystać z opcji Ghidry
która pozwala nam przekopiować dane już w formacie
tablicy w języku C



Dwoma kliknięciami mamy wszystko ładnie przekopiowane

```
{ 0x5d, 0x10, 0x14, 0x4c, 0x43, 0x10, 0x43, 0x4e, 0x4d, 0x14, 0x3f, 0x47, 0x4c, 0x34,
0x23, 0x26, 0x41, 0x5b, 0x28, 0x52, 0x10, 0x11, 0x3f, 0x53, 0x11, 0x4c, 0x54, 0x52,
0x00 }
```

Teraz rozumiając jak działa funkcja **generate_flag**
i wiemy do jakich danych musimy **"dojść"**
musimy odwrócić proces funkcji **generate_flag** by uzyskać input który
będzie pasował do funkcji **strcmp**

Stworzyłem funkcję w języku **C++**
zmienna **formated_flag to dane przekopiowane z programu**
original_flag to bedzie nasz docelowy input

musiałem odwrócić proces przestawiania literek
z uwzględnieniem że jak wtedy **odejmowała** funkcja wartość **0x20**
od każdego znaku
to musimy **dodać wartość 0x20** do każdego znaku
Żeby łatwiej zrozumieć to opiszę to na krótkim testowym inpucie
(proces dla funkcji w pliku .exe)

abcdef	taką wartość wpisaliśmy
ABCDEF	na taką zostanie zamieniona każda wartość
FDBACE	i tak zostanie pomieszana

(proces dla funkcji którą to odwracam)

FDBACE	taki byłby nasz output
fdbace	więc **zwiększymy** sobie to o wartość **0x20**
abcdef	i teraz musimy odwrócic kolejnośc sposobem"

sposob" na bazie tego jak działa funkcja pierwotna
polega na iterowaniu od początku naszego outputu i zapisywania go
na ostatnią pozycję i cofania się o -2 wzgledem indeksu
gdy dojdziemy do początku wtedy wpisujemy dalsze wartości
ale idąc do końca tablicy charów o 2 kroki

```
char formated_flag[] = { 0x5d, 0x10, 0x14, 0x4c, 0x43, 0x10, 0x43, 0x4e, 0x4d, 0x14, 0x3f, 0x47, 0x4c, 0
char original_flag[28] = {0x00};

input_length = strlen(formated_flag);

int j = 0;
for (int i=input_length-1; i>0; i=i-2) {
    original_flag[i] = formated_flag[j] + 0x20;
    j++;
}
for (int i = 0; i < input_length; i = i+2) {
    original_flag[i] = formated_flag[j] + 0x20;
    j++;
}

for (int i = 0; i < input_length; i = i + 1) {
    printf("%c", original_flag[i]);
}
printf("\n");


return 0;
}
```

```
Konsola debugowania programu Microsoft Visual Studio
CTFlag{_H4rm0n1c_0sc1ll4t0r}

C:\Users\judasz\source\repos\ConsoleApplication7\Debug
Aby automatycznie zamknąć konsolę po zatrzymaniu debug
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Otrzymujemy wtedy flagę
CTFlag{_H4rm0n1c_0sc1ll4t0r}

Jest to prawidlowa flaga! :D

```
┌──(kali㉿kali)-[~/Desktop/60d]
└─$ ./grandfather_clock CTFlag{_H4rm0n1c_0sc1ll4t0r}
Congratulations! You've cracked the code!

┌──(kali㉿kali)-[~/Desktop/60d]
└─$
```