Phase 2 report

SAVING THE WORLD

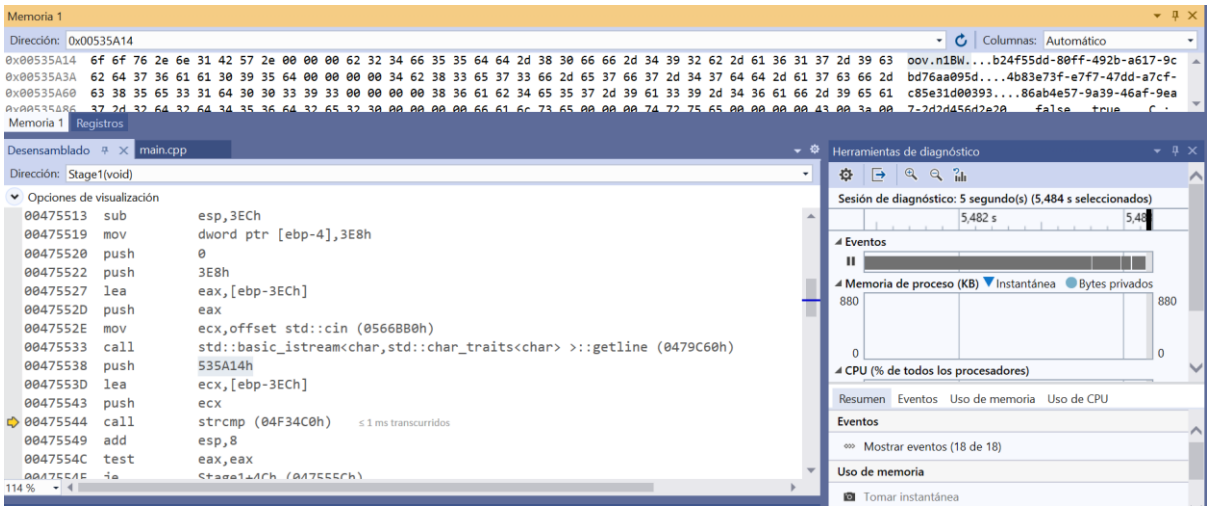
Mikel Fernández Esparta
Miguel Gayol Pertierra
Jorge Joaquín Gancedo Fernández

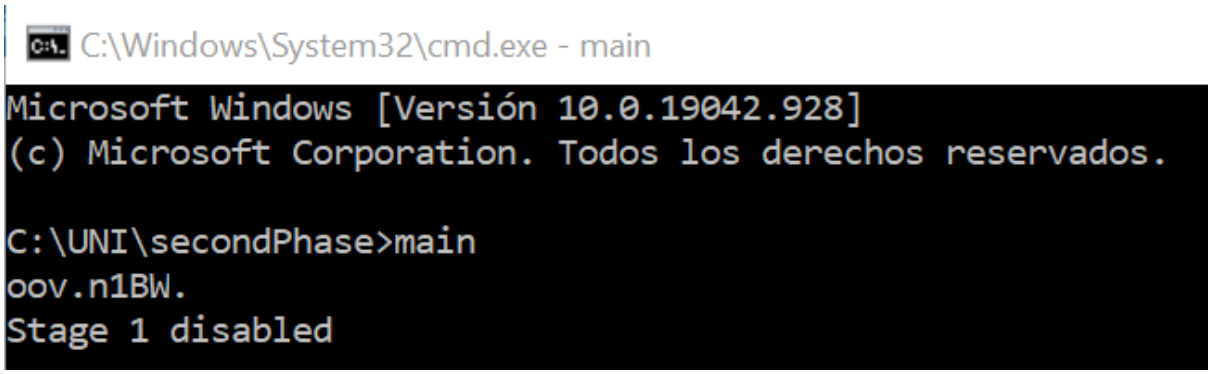
Obtaining each bomb's password

Bomb 1

After requesting a password to the user, in the code we can see tet it pushes 535A14h, in this memory location happens to be the actual password that we have to type in in order to disable the first stage, which will later on be compared with the one that we have written before. This password must be 9 digits long, so if we access the memory location, we select the first nine digits as you can see in the following screenshot highligthed.

Otherwise, the following message is printed "Oh, no, the world is over! BOOM!".





Bomb 2

The Stage 2 consists in adding a total of 3 numeric inputs. The clue to solve this problem is to make sure the subtraction of the first and second inputs is the total number of 4. The third numeric input is irrelevant.

In the picture below we see in the red box the main clue to the issue, [ebx] represents the first input and [ebx + 4] the second one, then there is a comparison where if the subtraction between [ebx] and [ebx + 4] is not 4, will jump to the instruction 'Explode' which will destroy the world.

```
004755A3  call    std::basic_istream<char,std::char_tr
004755A8  jmp     Stage2+17h (0475587h)
004755AA  mov     dword ptr [ebp-8],1
004755B1  lea     ebx,[ebp-18h]
004755B4  mov     eax,dword ptr [ebx]
004755B6  sub     eax,dword ptr [ebx+4]     ≤1 ms transcurric
004755B9  cmp     eax,4
004755BC  jne     Stage2+55h (04755C5h)
004755BE  mov     dword ptr [ebp-8],0
004755C5  cmp     dword ptr [ebp-8],0
004755C9  je      Stage2+67h (04755D7h)
004755CB  push    2
004755CD  call    Explode (0475450h)
004755D2  add     esp,4
004755D5  jmp     Stage2+76h (04755E6h)
004755D7  push    535A48h
004755DC  push    2
004755DE  call    Defuse (04754C0h)
004755E3  add     esp,8
004755E6  pop     ebx
004755E7  mov     esp,ebp
004755E9  pop     ebp
```

Unsuccessful cases:

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
Hola que tal
Oh, no, the world is over! BOOM!

D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>
```

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
-1
-6
5
Oh, no, the world is over! BOOM!
```

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
9
3
9
Oh, no, the world is over! BOOM!
```

Successful cases:

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
17
13
56
Stage 2 disabled
```

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
103
99
99999999
Stage 2 disabled
```

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
-5
-9
8
Stage 2 disabled
```

Bomb 3

Bomb 3 is always deactivated with a string value given.

```
004755F0 55                    push        ebp
004755F1 8B EC                 mov         ebp,esp
004755F3 83 EC 14              sub         esp,14h
004755F6 8D 45 F8              lea         eax,[ebp-8]
004755F9 50                    push        eax
004755FA 8D 4D FC              lea         ecx,[ebp-4]
004755FD 51                    push        ecx
004755FE B9 B0 6B 56 00        mov         ecx,offset std::cin (0566BB0h)
00475603 E8 68 0F 00 00        call        std::basic_istream<char,std::char_traits<char> >::operator>> (0476570h)
00475608 8B C8                 mov         ecx,eax
0047560A E8 61 0F 00 00        call        std::basic_istream<char,std::char_traits<char> >::operator>> (0476570h)
0047560F 8B 55 F8              mov         edx,dword ptr [ebp-8]    ≤1 ms transcurridos
00475612 81 E2 00 01 00 00     and         edx,100h
00475618 C1 FA 08              sar         edx,8
0047561B 89 55 F4              mov         dword ptr [ebp-0Ch],edx
0047561E 8B 45 FC              mov         eax,dword ptr [ebp-4]
00475621 25 00 00 02 00        and         eax,20000h
00475626 C1 F8 11              sar         eax,11h
00475629 89 45 F0              mov         dword ptr [ebp-10h],eax
0047562C 8B 4D FC              mov         ecx,dword ptr [ebp-4]
0047562F 81 E1 00 00 10 00     and         ecx,100000h
00475635 C1 F9 14              sar         ecx,14h
00475638 89 4D EC              mov         dword ptr [ebp-14h],ecx
0047563B 8B 55 F4              mov         edx,dword ptr [ebp-0Ch]
0047563E 3B 55 F0              cmp         edx,dword ptr [ebp-10h]
00475641 75 06                 jne         Stage3+59h (0475649h)
00475643 83 7D EC 00           cmp         dword ptr [ebp-14h],0
00475647 75 0C                 jne         Stage3+65h (0475655h)
00475649 6A 03                 push        3
0047564B E8 00 FE FF FF        call        Explode (0475450h)
00475650 83 C4 04              add         esp,4
00475653 EB 0F                 jmp         Stage3+74h (0475664h)
00475655 68 70 5A 53 00        push        535A70h
0047565A 6A 03                 push        3
0047565C E8 5F FE FF FF        call        Defuse (04754C0h)
00475661 83 C4 08              add         esp,8
00475664 8B E5                 mov         esp,ebp
00475666 5D                    pop         ebp
```

Every time a string is given as input, the jump on address line 00475647h sends the program to the line 00475655h, defusing the last step.

Unsuccessful cases:

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
5
1
1
Stage 2 disabled
500
0
Oh, no, the world is over! BOOM!
```

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
4
0
0
Stage 2 disabled
7777
string
Oh, no, the world is over! BOOM!
```

Successful cases:

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
4
0
0
Stage 2 disabled
string
Stage 3 disabled
Wow, you've just saved the Earth!
```

```
D:\Universidad\ASIGNATURAS\FCR\2021\Teamwork\secondPhase>main
oov.n1BW.
Stage 1 disabled
4
0
0
Stage 2 disabled
hola
Stage 3 disabled
Wow, you've just saved the Earth!
```

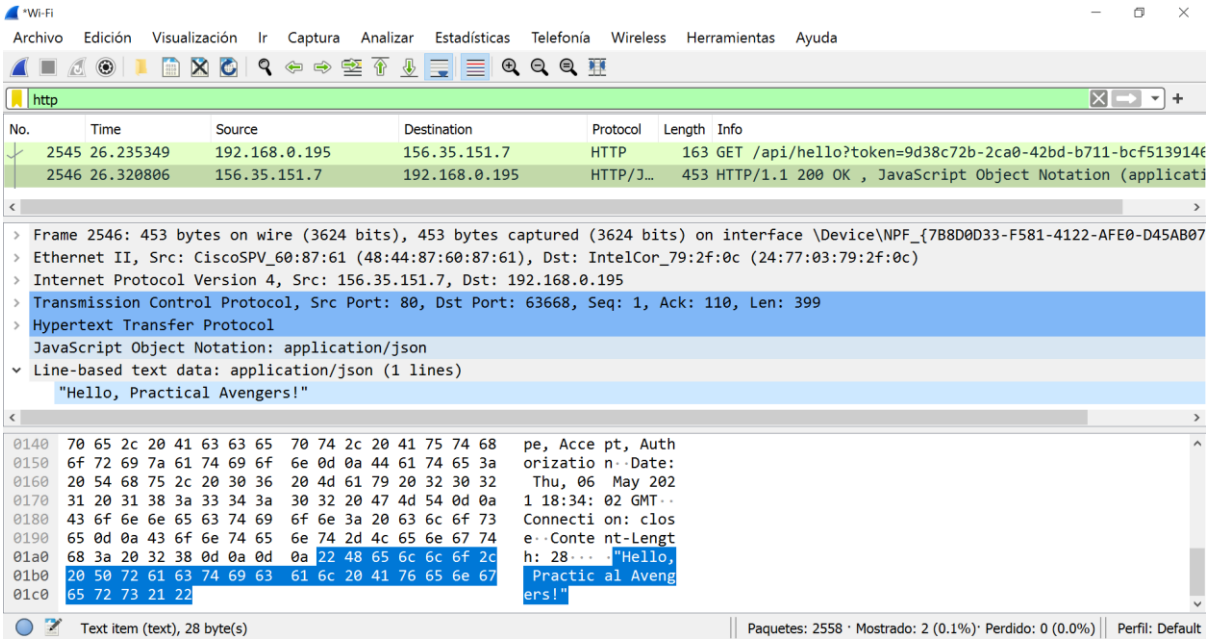
Patching the .exe

To patch the .exe file, we first make a copy of it to use it with HxD while looking at the code in the Visual Studio Disassembler.

- Stage 1 was patched copying the code that is executed when a successful input is given and pasting it at the top of the function that compares the input and the password of the bomb.
- Stage 2 was patched pasting a *jmp* that goes to a *call* to a function named *Defuse()* and later produces a *ret*. Therefore, any input is accepted.
- In stage 3, *ret* was put instead of the original code, so it always deactivates with no inputs.

Name of the subgroup

By using a network analyser, in this case Wireshark, we connect to our Wi-Fi. Then we need to execute the main.exe, once done we need to stop the capture of packages, the red button on the left-upper corner. Afterwards filter HTTP protocols and we can the find name of the criminal subgroup “Practical Avengers” on the Line-based text data. We can also check the encoding in hexadecimal.



Work division

Just like with the first phase, we decided to work all together via Teams, during the last two weeks. On one of our first meetings, Mikel and Miguel were able to defuse all the bombs, while Jorge was working on modifying the code to defuse all the bombs, even with invalid inputs. Later, we would all work on the report explaining how we got the results and taking the necessary screenshots of the valid and invalid inputs. The thing that took the most time was probably modifying the code, but once it worked and we saw how Wireshark worked we discovered the name of the criminal subgroup.