Category:

Reversing

Name:

Bad Tool

Message:

You are provided with an executable file named “BadTool.exe”. This file is used by notorious hacking group for their malicious activities. This file is believed to contain crucial evidence that could lead to the group’s downfall.

However, the file is protected by a password, and without it, the investigations cannot go further. Find out the password hidden within this file. The flag for this challenge is displayed if you enter the correct password.

Hints:

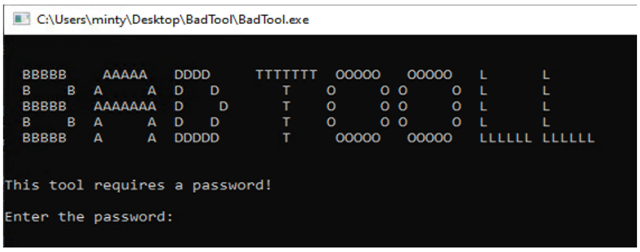
- If you're using the reverse engineering approach, the first step is to identify the function responsible for verifying the authenticity of the input. This function handles the comparison between the user-provided password and the correct one.

Objective:

Your task is to reveal the flag from “BadTool.exe” by entering the correct password. This requires fundamental reversing technique.

Instructions:

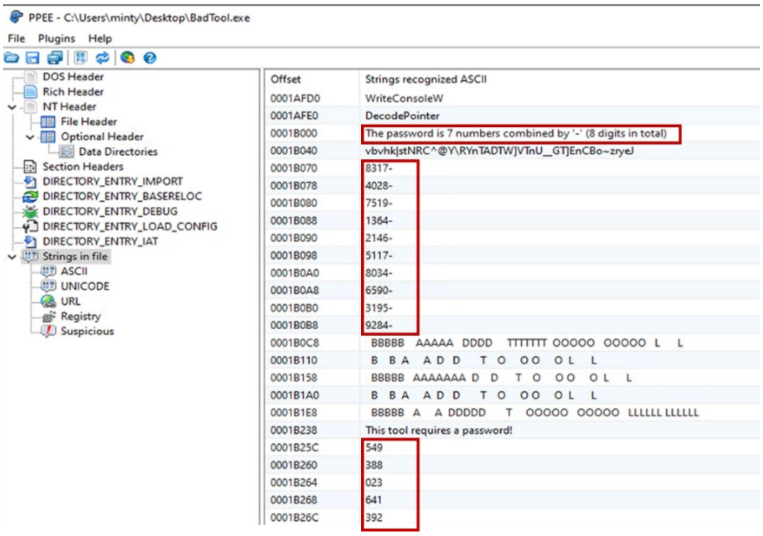
1. Download the zip file (“BadTool.zip”) and extract it to get the executable file “BadTool.exe”. No password is required to extract. Once executed, it asks for the password.



```
C:\Users\minty\Desktop\BadTool\BadTool.exe

  BBBBB    AAAAA   DDDD     TTTTTTT  OOOOO   OOOOO   L      L
  B    B  A     A  D   D       T    O     O O     O  L      L
  BBBBB   AAAAAAA  D    D      T    O     O O     O  L      L
  B    B  A     A  D   D       T    O     O O     O  L      L
  BBBBB   A     A  DDDDD       T     OOOOO   OOOOO   LLLLLL LLLLLL

This tool requires a password!

Enter the password:
```

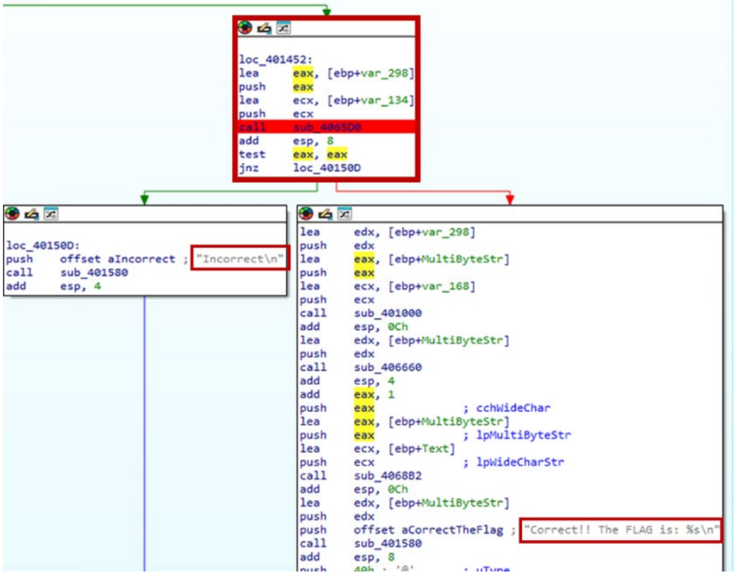
2. Start from checking the file by using executable file analysis tool. The tool explained here is “PeStudio”, however any executable file analyzing tool should work as well. Open the file on PeStudio. Upon examining the list of strings, you can identify messages corresponding to incorrect password attempts ("Incorrect") and successful ones ("Correct!! The FLAG is: %s").

Additionally, there is a hint indicating the password format as a 7-digit sequence separated by a hyphen ("-"). Several dummy 4-digit and 3-digit numbers are also present within the string list.



Offset	Strings recognized ASCII
0001AFD0	WriteConsoleW
0001AFE0	DecodePointer
0001B000	The password is 7 numbers combined by '-' (8 digits in total)
0001B040	vbvhk|stNRC^@Y\RYnTADTW]VTnU__GT]EnCBo~zryeJ
0001B070	8317-
0001B078	4028-
0001B080	7519-
0001B088	1364-
0001B090	2146-
0001B098	5117-
0001B0A0	8034-
0001B0A8	6590-
0001B0B0	3195-
0001B0B8	9284-
0001B0C8	BBBBB AAAAA DDDD TTTTTTT OOOOO OOOOO L L
0001B110	B B A A D D T O OO O L L
0001B158	BBBBB AAAAAAA D D T O OO O L L
0001B1A0	B B A A D D T O OO O L L
0001B1E8	BBBBB A A DDDDD T OOOOO OOOOO LLLLLL LLLLLL
0001B238	This tool requires a password!
0001B25C	549
0001B260	388
0001B264	023
0001B268	641
0001B26C	392

3. While a brute force approach could be used to determine the correct combination, we'll save time by using reverse engineering techniques instead. Start by opening "BadTool.exe" in IDA Free and navigate to the graph view. From there, locate the section of code related to the flag-revealing process. Scroll down to identify the highlighted functions as shown below.



```
loc_401452:
lea     eax, [ebp+var_298]
push    eax
lea     ecx, [ebp+var_134]
push    ecx
call    sub_4065D0
add     esp, 8
test    eax, eax
jnz     loc_40150D
```

```
loc_40150D:
push    offset aIncorrect ; "Incorrect\n"
call    sub_401580
add     esp, 4
```

```
lea     edx, [ebp+var_298]
push    edx
lea     eax, [ebp+MultiByteStr]
push    eax
lea     ecx, [ebp+var_168]
push    ecx
call    sub_401000
add     esp, 0Ch
lea     edx, [ebp+MultiByteStr]
push    edx
call    sub_406660
add     esp, 4
add     eax, 1
push    eax             ; cchWideChar
lea     eax, [ebp+MultiByteStr]
push    eax             ; lpMultiByteStr
lea     ecx, [ebp+Text]
push    ecx             ; lpWideCharStr
call    sub_4068B2
add     esp, 0Ch
lea     edx, [ebp+MultiByteStr]
push    edx
push    offset aCorrectTheFlag ; "Correct!! The FLAG is: %s\n"
call    sub_401580
add     esp, 8
push    40h ; '@'       ; uType
```

4. The block outlined with the bold red line is responsible for determining whether the provided password is correct. This block contains the function that compares the user input with the correct password. Set a breakpoint at the function call, sub_XXXXXX (depends on the execution environment), and run the debugger. After pressing [F9] twice, the debugger will pause and wait for password input. Enter a random string (e.g., "0000-000") into the running "BadTool.exe". The debugger will halt at the breakpoint. By inspecting the values in the

registers and memory, you'll observe that the function uses the eax and ecx registers, with eax containing "5117-023" and ecx holding the user input ("0000-000" in this case). This confirms that the function is comparing the entered value with the correct password, revealing that "5117-023" is the correct password.

5. Return to the “BadTool.exe” and enter the obtained password. Now you should see the flag!

Flag is:
CSG_FLAG{criminal_evidence_brought_to_LIGHT }

References:

● PeStudio	https://www.winitor.com/download
● PPEE (Puppy)	https://mzrst.com/
● Detect It Easy (DIE)	https://github.com/horsicq/Detect-It-Easy