BABY RE

Category: Reverse Engineer

Difficulty: Easy

Creator: Xh4H

Challenge Description: “Show us your basic skills! (P.S. There are 4 ways to solve this, are you willing to try them all?)”

Download and unzip the archive to retrieve the file. Analyzing with the `file` command show that this is a Linux executable.

```
┌─[parrot@parrot]─[~/Desktop]
└──╼ $file baby
baby: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked
, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=25adc53b89f781335a27bf1
b81f5c4cb74581022, for GNU/Linux 3.2.0, not stripped
```

If we make the file executable with `chmod +x` and run it, it asks for a key to be inserted. Simply hitting enter gives us the message “Try again later.”

```
┌─[parrot@parrot]─[~/Desktop]
└──╼ $./baby
Insert key:

Try again later.
```

The Easy Way

Running `hexdump -C` shows the readable contents of the file.

```
┌─[parrot@parrot]─[~/Desktop]
└──╼ $hexdump -C baby
```

```
000011a0  c0 75 37 48 b8 48 54 42  7b 42 34 42 59 48 ba 5f  |.u7H.HTB{B4BYH._|
000011b0  52 33 56 5f 54 48 34 48  89 45 c0 48 89 55 c8 c7  |R3V_TH4H.E.H.U..|
000011c0  45 d0 54 53 5f 45 66 c7  45 d4 5a 7d 48 8d 45 c0  |E.TS_Ef.E.Z}H.E.|
000011d0  48 89 c7 e8 58 fe ff ff  eb 0c 48 8d 3d 7f 0e 00  |H...X.....H.=...|
000011e0  00 e8 4a fe ff ff b8 00  00 00 00 c9 c3 0f 1f 00  |..J.............|
000011f0  41 57 4c 8d 3d ef 2b 00  00 41 56 49 89 d6 41 55  |AWL.=.+..AVI..AU|
00001200  49 89 f5 41 54 41 89 fc  55 48 8d 2d e0 2b 00 00  |I..ATA..UH.-.+..|
00001210  53 4c 29 fd 48 83 ec 08  e8 e3 fd ff ff 48 c1 fd  |SL).H........H..|
00001220  03 74 1b 31 db 0f 1f 00  4c 89 f2 4c 89 ee 44 89  |.t.1....L..L..D.|
00001230  e7 41 ff 14 df 48 83 c3  01 48 39 dd 75 ea 48 83  |.A...H...H9.u.H.|
00001240  c4 08 5b 5d 41 5c 41 5d  41 5e 41 5f c3 0f 1f 00  |..[]A\A]A^A_....|
00001250  c3 00 00 00 48 83 ec 08  48 83 c4 08 c3 00 00 00  |....H...H.......|
00001260  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00002000  01 00 02 00 00 00 00 00  44 6f 6e 74 20 72 75 6e  |........Dont run|
00002010  20 60 73 74 72 69 6e 67  73 60 20 6f 6e 20 74 68  | `strings` on th|
00002020  69 73 20 63 68 61 6c 6c  65 6e 67 65 2c 20 74 68  |is challenge, th|
00002030  61 74 20 69 73 20 6e 6f  74 20 74 68 65 20 77 61  |at is not the wa|
00002040  79 21 21 21 21 00 49 6e  73 65 72 74 20 6b 65 79  |y!!!!.Insert key|
00002050  3a 20 00 61 62 63 64 65  31 32 32 33 31 33 0a 00  |: .abcde122313..|
00002060  54 72 79 20 61 67 61 69  6e 20 6c 61 74 65 72 2e  |Try again later.|
```

Within the contents of the file is what appears to be a flag that is not quite properly formatted, a note saying, “Don’t run `strings` on this challenge, that is not the way!!!!”. There is also the messages “Insert key” and “Try again later” with the value “abcde122313” nested between. Let’s try adding entering that as the key.

```
Insert key:
abcde122313
HTB{B4BY_R3V_TH4TS_EZ}
```

Success! Naturally we could have found the same information using `strings`.

```
HTB{B4BYH
_R3V_TH4H
_TS_Ef
[]A\A]A^A_
Dont run `strings` on this challenge, that is not the way!!!!
Insert key:
abcde122313
Try again later.
```

The Harder Way

We can take a closer look at how the flag is being created by using radare2, analyzing the file with `aaa`, then disassembling main.

```
┌─[parrot@parrot]─[~/Desktop]
└──╼ $r2 baby
[0x00001070]> aaa
```

```
[0x00001070]> pd @main
            ; DATA XREF from entry0 @ 0x108d
┌ 152: int main (int argc, char **argv, char **envp);
            ; var char *s @ rbp-0x40
            ; var int64_t var_38h @ rbp-0x38
            ; var int64_t var_30h @ rbp-0x30
            ; var int64_t var_2ch @ rbp-0x2c
            ; var char *s1 @ rbp-0x20
            ; var char *var_8h @ rbp-0x8
            0x00001155      55              push rbp
            0x00001156      4889e5          mov rbp, rsp
            0x00001159      4883ec40        sub rsp, 0x40
            0x0000115d      488d05a40e00.   lea rax, qword str.Dont_run__strings__on_this_challenge__that_is_not_the_way
; 0x2008 ; "Dont run `strings` on this challenge, that is not the way!!!!"
            0x00001164      488945f8        mov qword [var_8h], rax
            0x00001168      488d3dd70e00.   lea rdi, qword str.Insert_key: ; 0x2046 ; "Insert key: " ; const char *s
            0x0000116f      e8bcfeffff      call sym.imp.puts               ; int puts(const char *s)
            0x00001174      488b15c52e00.   mov rdx, qword [obj.stdin]      ; rdi
                                                                            ; [0x4040:8]=0 ; FILE *stream
            0x0000117b      488d45e0        lea rax, qword [s1]
            0x0000117f      be14000000      mov esi, 0x14                   ; int size
            0x00001184      4889c7          mov rdi, rax                    ; char *s
            0x00001187      e8b4feffff      call sym.imp.fgets              ; char *fgets(char *s, int size, FILE *stream)
            0x0000118c      488d45e0        lea rax, qword [s1]
            0x00001190      488d35bc0e00.   lea rsi, qword str.abcde122313 ; 0x2053 ; "abcde122313\n" ; const char *s2
            0x00001197      4889c7          mov rdi, rax                    ; const char *s1
            0x0000119a      e8b1feffff      call sym.imp.strcmp             ; int strcmp(const char *s1, const char *s2)
            0x0000119f      85c0            test eax, eax
        ┌─< 0x000011a1      7537            jne 0x11da
        │   0x000011a3      48b84854427b.   movabs rax, 0x594234427b425448 ; 'HTB{B4BY'
        │   0x000011ad      48ba5f523356.   movabs rdx, 0x3448545f5633525f
        │   0x000011b7      488945c0        mov qword [s], rax
        │   0x000011bb      488955c8        mov qword [var_38h], rdx
        │   0x000011bf      c745d054535f.   mov dword [var_30h], 0x455f5354 ; 'TS_E'
        │   0x000011c6      66c745d45a7d    mov word [var_2ch], 0x7d5a  ; 'Z}'
        │   0x000011cc      488d45c0        lea rax, qword [s]
        │   0x000011d0      4889c7          mov rdi, rax                    ; const char *s
        │   0x000011d3      e858feffff      call sym.imp.puts               ; int puts(const char *s)
```

The important pieces here are the variables on the stack…

```
; var char *s @ rbp-0x40
; var int64_t var_38h @ rbp-0x38
; var int64_t var_30h @ rbp-0x30
; var int64_t var_2ch @ rbp-0x2c
; var char *s1 @ rbp-0x20
; var char *var_8h @ rbp-0x8
```

…and what looks to be flag construction and output.

```
0x000011a3      48b84854427b.   movabs rax, 0x594234427b425448 ; 'HTB{B4BY'
0x000011ad      48ba5f523356.   movabs rdx, 0x3448545f5633525f
0x000011b7      488945c0        mov qword [s], rax
0x000011bb      488955c8        mov qword [var_38h], rdx
0x000011bf      c745d054535f.   mov dword [var_30h], 0x455f5354 ; 'TS_E'
0x000011c6      66c745d45a7d    mov word [var_2ch], 0x7d5a  ; 'Z}'
0x000011cc      488d45c0        lea rax, qword [s]
0x000011d0      4889c7          mov rdi, rax                    ; const char *s
0x000011d3      e858feffff      call sym.imp.puts               ; int puts(const char *s)
```

The break down of the code is as follows:

1. (0x11a3) The value “HTB{B4BY” is loaded into the register RAX.
2. (0x11ad) Radare2 doesn’t give a nice print out here, but if we split the values into hexadecimal bytes, reverse the order to account for little-endianness of x86 systems, and translate to an ASCII character, we can see that this line is storing the value “_R3V_TH4” into the register RDX.

3. (0x11b7) “HTB{B4BY”, stored in RAX, is moved into the character pointer at the address referenced by RBP-0x40.
4. (0x11bb) “_R3V_TH4”, stored in RDX, is moved into var_38h located at RBP-0x38.
5. (0x11bf) “TS_E” is moved into var_30h at the address of rbp-0x30.
6. (0x11c6) “Z}” is moved into var_2ch at the address of rbp-0x2c.
7. (0x11cc – 0x11d0) Registers are loaded with our values in preparation for a system call.
8. (0x11d3) Puts() is called to print the value of our string located in the range of rbp-0x40 through rbp-0x20.

To simplify, this code simply constructs the string “HTB{B4BY_TH4TS_EZ}” and prints it to stdout. With this information we can by pass entering the key altogether!