Port:

```
ssh -p 2226 narnia1@narnia.labs.overthewire.org
```

Level 0 Source Code:

```
int main(){
    long val=0x41414141; --> hexadecimal meaning AAAA
    char buf[20]; --> character array of size 20; intended to store user input.

    printf("Correct val's value from 0x41414141 -> 0xdeadbeef!\n"); --> prompts the user to change
the value of AAAA to 0xdeadbeef.
    printf("Here is your chance: ");
    scanf("%24s",&buf); --> %24s reads a strings up to 24 char from the user and stores it in buf.
However, buf is only 20 char (=/= 24). So if user enters more than 20, this causes buffer overflow, this
allows overwriting of adjacent memory, including value of val. So you can carefully overwrite the
memroy in val in order to set it equal to 0xdeadbeef.

    printf("buf: %s\n",buf);
    printf("val: 0x%08x\n",val);

    if(val==0xdeadbeef){
        setreuid(geteuid(),geteuid());
        system("/bin/sh"); --> spawns shell as well as upgrades privileges
    }
    else {
        printf("WAY OFF!!!!\n");
        exit(1);
    }

    return 0;
}
```

Level 1 Source Code:

```
int main(){
    int (*ret)();

    if(getenv("EGG")==NULL){
        printf("Give me something to execute at the env-variable EGG\n");
        exit(1);
    }

    printf("Trying to execute EGG!\n");
    ret = getenv("EGG");
    ret();
    return 0;
}
```

Level 1 Assembly Code:

```
Dump of assembler code for function main:
   0x08049186 <+0>:push   ebp
   0x08049187 <+1>:mov    ebp,esp
   0x08049189 <+3>:sub    esp,0x4
   0x0804918c <+6>:push   0x804a008
   0x08049191 <+11>:	call   0x8049040 <getenv@plt>
   0x08049196 <+16>:	add    esp,0x4
   0x08049199 <+19>:	test   eax,eax
   0x0804919b <+21>:	jne    0x80491b1 <main+43>
   0x0804919d <+23>:	push   0x804a00c
   0x080491a2 <+28>:	call   0x8049050 <puts@plt>
   0x080491a7 <+33>:	add    esp,0x4
   0x080491aa <+36>:	push   0x1
   0x080491ac <+38>:	call   0x8049060 <exit@plt>
   0x080491b1 <+43>:	push   0x804a041
   0x080491b6 <+48>:	call   0x8049050 <puts@plt>
   0x080491bb <+53>:	add    esp,0x4
   0x080491be <+56>:	push   0x804a008
   0x080491c3 <+61>:	call   0x8049040 <getenv@plt>
   0x080491c8 <+66>:	add    esp,0x4
   0x080491cb <+69>:	mov    DWORD PTR [ebp-0x4],eax
   0x080491ce <+72>:	mov    eax,DWORD PTR [ebp-0x4]
   0x080491d1 <+75>:	call   eax
   0x080491d3 <+77>:	mov    eax,0x0
   0x080491d8 <+82>:	leave
   0x080491d9 <+83>:	ret
End of assembler dump.
```

Level 2 Code:

```
int main(int argc, char * argv[]){
    char buf[128];

    if(argc == 1){
        printf("Usage: %s argument\n", argv[0]);
```
--> if no argument is provided, it will just spit out that it requires an argument and then the program will close

```
        exit(1);
    }
    strcpy(buf,argv[1]);
```
--> we see here that it will copy the first command-line argument into the character array buf. Buf buf only has 128 bytes, and strcpy (string copy) does not perform bound

checking and thus this hints at buffer overflow.

```
    printf("%s", buf);

    return 0;
}
```

Level 2 Disassembly Code:

- 0x08049186 <+0>:   push   ebp
- 0x08049187 <+1>:mov   ebp,esp
- 0x08049189 <+3>:add   esp,0xffffff80
- 0x0804918c <+6>:cmp   DWORD PTR [ebp+0x8],0x1
- 0x08049190 <+10>:   jne   0x80491ac <main+38>
- 0x08049192 <+12>:   mov   eax,DWORD PTR [ebp+0xc]
- 0x08049195 <+15>:   mov   eax,DWORD PTR [eax]
- 0x08049197 <+17>:   push   eax
- 0x08049198 <+18>:   push   0x804a008
- 0x0804919d <+23>:   call   0x8049040 <printf@plt>
- 0x080491a2 <+28>:   add   esp,0x8
- 0x080491a5 <+31>:   push   0x1
- 0x080491a7 <+33>:   call   0x8049060 <exit@plt>
- 0x080491ac <+38>:   mov   eax,DWORD PTR [ebp+0xc]
- 0x080491af <+41>:   add   eax,0x4
- 0x080491b2 <+44>:   mov   eax,DWORD PTR [eax]
- 0x080491b4 <+46>:   push   eax
- 0x080491b5 <+47>:   lea   eax,[ebp-0x80]
- 0x080491b8 <+50>:   push   eax
- 0x080491b9 <+51>:   call   0x8049050 <strcpy@plt>
- 0x080491be <+56>:   add   esp,0x8
- 0x080491c1 <+59>:   lea   eax,[ebp-0x80]
- 0x080491c4 <+62>:   push   eax
- 0x080491c5 <+63>:   push   0x804a01c
- 0x080491ca <+68>:   call   0x8049040 <printf@plt>
- 0x080491cf <+73>:   add   esp,0x8
- 0x080491d2 <+76>:   mov   eax,0x0
- 0x080491d7 <+81>:   leave
- 0x080491d8 <+82>:   ret

Description:

- Allocates 128 bytes on the stack for the buf array by subtracting 0x80 from esp (note 0xffffff80 = 128 in decimal)
- cmp means compare with 0x1 (conditional), if the argument is not 0x1, then it will jump to +38
- We can see before +38 that the conditional has <printf> and <exit> for that conditional section.
- For main +38, we see <strcpy> happens at +51, loading the buf argument [ebp-0x8] onto eax. Then calls <printf> to print contents of buf.