Level14

This is the final level. After some long enumeration with no juicy finds, the option looks like we are only left with the exploitation of the `getflag` binary. Let's download it on our machine with the `scp` command and run it trough **Ghidra**'s code browser to see how it works.

```
undefined4 main(void)
{
  bool bVar1;
  FILE *__stream;
  long lVar2;
  undefined4 uVar3;
  char *pcVar4;
  int iVar5;
  __uid_t _Var6;
  int iVar7;
  int in_GS_OFFSET;
  undefined local_114 [256];
  int local_14;

  local_14 = *(int *)(in_GS_OFFSET + 0x14);
  bVar1 = false;
  lVar2 = ptrace(PTRACE_TRACEME,0,1,0);
  if (lVar2 < 0) {
    puts("You should not reverse this");
    uVar3 = 1;
  }
  else {
    pcVar4 = getenv("LD_PRELOAD");
    if (pcVar4 == (char *)0x0) {
      iVar5 = open("/etc/ld.so.preload",0);
      if (iVar5 < 1) {
        iVar5 = syscall_open("/proc/self/maps",0);
        if (iVar5 == -1) {
          fwrite("/proc/self/maps is unaccessible, probably a LD_PRELOAD
attempt exit..\n",1,0x46,
                 stderr);
          uVar3 = 1;
        }
        else {
          do {
            do {
              while( true ) {
                iVar7 = syscall_gets(local_114,0x100,iVar5);
                if (iVar7 == 0) goto LAB_08048ead;
                iVar7 = isLib(local_114,&DAT_08049063);
                if (iVar7 == 0) break;
                bVar1 = true;
```

```
            }
          } while (!bVar1);
          iVar7 = isLib(local_114,&DAT_08049068);
          if (iVar7 != 0) {
            fwrite("Check flag.Here is your token : ",1,0x20,stdout);
            _Var6 = getuid();
            __stream = stdout;
            if (_Var6 == 0xbbe) {
              pcVar4 = (char *)ft_des("H8B8h_20B4J43><8>\\ED<;j@3");
              fputs(pcVar4,__stream);
            }
            else if (_Var6 < 0xbbf) {
              if (_Var6 == 0xbba) {
                pcVar4 = (char *)ft_des("<>B16\\AD<C6,G_<1>^7ci>l4B");
                fputs(pcVar4,__stream);
              }
              else if (_Var6 < 0xbbb) {
                if (_Var6 == 3000) {
                  pcVar4 = (char *)ft_des("I`fA>_88eEd:=`85h0D8HE>,D");
                  fputs(pcVar4,__stream);
                }
                else if (_Var6 < 0xbb9) {
                  if (_Var6 == 0) {
                    fwrite("You are root are you that dumb ?
\n",1,0x21,stdout);
                  }
                  else {
LAB_08048e06:
                    fwrite("\nNope there is no token here for you sorry. Try
again :)",1,0x38,
                           stdout);
                  }
                }
                else {
                  pcVar4 = (char *)ft_des("7`4Ci4=^d=J,?>i;6,7d416,7");
                  fputs(pcVar4,__stream);
                }
              }
              else if (_Var6 == 0xbbc) {
                pcVar4 = (char *)ft_des("?4d@:,C>8C60G>8:h:Gb4?l,A");
                fputs(pcVar4,__stream);
              }
              else if (_Var6 < 0xbbd) {
                pcVar4 = (char *)ft_des("B8b:6,3fj7:,;bh>D@>8i:6@D");
                fputs(pcVar4,__stream);
              }
              else {
                pcVar4 = (char *)ft_des("G8H.6,=4k5J0<cd/D@>>B:>:4");
                fputs(pcVar4,__stream);
              }
            }
            else if (_Var6 == 0xbc2) {
              pcVar4 = (char *)ft_des("74H9D^3ed7k05445J0E4e;Da4");
```

```
                fputs(pcVar4,__stream);
              }
              else if (_Var6 < 0xbc3) {
                if (_Var6 == 0xbc0) {
                  pcVar4 = (char *)ft_des("bci`mC{)jxkn<\"uD~6%g7FK`7");
                  fputs(pcVar4,__stream);
                }
                else if (_Var6 < 0xbc1) {
                  pcVar4 = (char *)ft_des("78H:J4<4<9i_I4k0J^5>B1j`9");
                  fputs(pcVar4,__stream);
                }
                else {
                  pcVar4 = (char *)ft_des("Dc6m~;}f8Cj#xFkel;#&ycfbK");
                  fputs(pcVar4,__stream);
                }
              }
              else if (_Var6 == 0xbc4) {
                pcVar4 = (char *)ft_des("8_Dw\"4#?+3i]q&;p6 gtw88EC");
                fputs(pcVar4,__stream);
              }
              else if (_Var6 < 0xbc4) {
                pcVar4 = (char *)ft_des("70hCi,E44Df[A4B/J@3f<=:`D");
                fputs(pcVar4,__stream);
              }
              else if (_Var6 == 0xbc5) {
                pcVar4 = (char *)ft_des("boe]!ai0FB@.:|L6l@A?>qJ}I");
                fputs(pcVar4,__stream);
              }
              else {
                if (_Var6 != 0xbc6) goto LAB_08048e06;
                pcVar4 = (char *)ft_des("g <t61:|4_|!@IF.-62FH&G~DCK/Ekrvvdwz?
v|");
                fputs(pcVar4,__stream);
              }
              fputc(10,stdout);
              goto LAB_08048ead;
            }
            iVar7 = afterSubstr(local_114,"00000000 00:00 0");
          } while (iVar7 != 0);
          fwrite("LD_PRELOAD detected through memory maps exit
..\n",1,0x30,stderr);
LAB_08048ead:
          uVar3 = 0;
        }
      }
      else {
        fwrite("Injection Linked lib detected exit..\n",1,0x25,stderr);
        uVar3 = 1;
      }
    }
    else {
      fwrite("Injection Linked lib detected exit..\n",1,0x25,stderr);
      uVar3 = 1;
```

```
    }
  }
  if (local_14 == *(int *)(in_GS_OFFSET + 0x14)) {
    return uVar3;
  }
                    /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}
```

The main function simply gets the **UID** of the user who called it and outputs the proper flag. All flags are present but they seem cyphered and passed trough a `ft_des()` function. We should take a look at that function.

```
char * ft_des(char *param_1)
{
  char cVar1;
  char *pcVar2;
  uint uVar3;
  char *pcVar4;
  byte bVar5;
  uint local_20;
  int local_1c;
  int local_18;
  int local_14;

  bVar5 = 0;
  pcVar2 = strdup(param_1);
  local_1c = 0;
  local_20 = 0;
  do {
    uVar3 = 0xffffffff;
    pcVar4 = pcVar2;
    do {
      if (uVar3 == 0) break;
      uVar3 = uVar3 - 1;
      cVar1 = *pcVar4;
      pcVar4 = pcVar4 + (uint)bVar5 * -2 + 1;
    } while (cVar1 != '\0');
    if (~uVar3 - 1 <= local_20) {
      return pcVar2;
    }
    if (local_1c == 6) {
      local_1c = 0;
    }
    if ((local_20 & 1) == 0) {
      if ((local_20 & 1) == 0) {
        for (local_14 = 0; local_14 < "0123456"[local_1c]; local_14 = local_14
+ 1) {
          pcVar2[local_20] = pcVar2[local_20] + -1;
          if (pcVar2[local_20] == '\x1f') {
            pcVar2[local_20] = '~';
          }
```

```
        }
      }
    }
    else {
      for (local_18 = 0; local_18 < "0123456"[local_1c]; local_18 = local_18 +
1) {
        pcVar2[local_20] = pcVar2[local_20] + '\x01';
        if (pcVar2[local_20] == '\x7f') {
          pcVar2[local_20] = ' ';
        }
      }
    }
    local_20 = local_20 + 1;
    local_1c = local_1c + 1;
  } while( true );
}
```

This confirms our guess and we should be able to take this algorithm, edit it a bit so it compiles in to a program that takes any **flag present in main** as argument and outputs it uncyphered.

```
#include <stdio.h>
#include <string.h>
int main(int ac, char **av)
{
  char cVar1;
  char *pcVar2;
  unsigned int uVar3;
  char *pcVar4;
  unsigned int bVar5;
  unsigned int local_20;
  int local_1c;
  int local_18;
  int local_14;

  if (ac < 2)
    puts("One argument is requiered");
  bVar5 = 0;
  pcVar2 = strdup(av[1]);
  local_1c = 0;
  local_20 = 0;
  do {
    uVar3 = 0xffffffff;
    pcVar4 = pcVar2;
    do {
      if (uVar3 == 0) break;
      uVar3 = uVar3 - 1;
      cVar1 = *pcVar4;
      pcVar4 = pcVar4 + (unsigned int)bVar5 * -2 + 1;
    } while (cVar1 != '\0');
    if (~uVar3 - 1 <= local_20) {
      printf("flag is = %s\n", pcVar2);
```

```
      return 0;
    }
    if (local_1c == 6) {
      local_1c = 0;
    }
    if ((local_20 & 1) == 0) {
      if ((local_20 & 1) == 0) {
        for (local_14 = 0; local_14 < "0123456"[local_1c]; local_14 = local_14
+ 1) {
          pcVar2[local_20] = pcVar2[local_20] + -1;
          if (pcVar2[local_20] == '\x1f') {
            pcVar2[local_20] = '~';
          }
        }
      }
    }
    else {
      for (local_18 = 0; local_18 < "0123456"[local_1c]; local_18 = local_18 +
1) {
        pcVar2[local_20] = pcVar2[local_20] + '\x01';
        if (pcVar2[local_20] == '\x7f') {
          pcVar2[local_20] = ' ';
        }
      }
    }
    local_20 = local_20 + 1;
    local_1c = local_1c + 1;
  } while( 1 );
}
```

Compiling this program and executing it results in proper uncyphering of the flags. We are able to get our last `level14` flag! We can find the proper flag by examining the UID if the user `flag14` in `/etc/passwd` wich is **3014 or 0xbc6** in hex.

```
... SNIP ...
              else {
                if (_Var6 != 0xbc6) goto LAB_08048e06;
                pcVar4 = (char *)ft_des("g <t61:|4_|!@IF.-62FH&G~DCK/Ekrvvdwz?
v|");
                fputs(pcVar4,__stream);
              }
... SNIP ...
```

```
... SNIP ...
flag00:x:3000:3000::/home/flag/flag00:/bin/bash
flag01:42hDRfypTqqnw:3001:3001::/home/flag/flag01:/bin/bash
flag02:x:3002:3002::/home/flag/flag02:/bin/bash
flag03:x:3003:3003::/home/flag/flag03:/bin/bash
flag04:x:3004:3004::/home/flag/flag04:/bin/bash
flag05:x:3005:3005::/home/flag/flag05:/bin/bash
```

```
flag06:x:3006:3006::/home/flag/flag06:/bin/bash
flag07:x:3007:3007::/home/flag/flag07:/bin/bash
flag08:x:3008:3008::/home/flag/flag08:/bin/bash
flag09:x:3009:3009::/home/flag/flag09:/bin/bash
flag10:x:3010:3010::/home/flag/flag10:/bin/bash
flag11:x:3011:3011::/home/flag/flag11:/bin/bash
flag12:x:3012:3012::/home/flag/flag12:/bin/bash
flag13:x:3013:3013::/home/flag/flag13:/bin/bash
flag14:x:3014:3014::/home/flag/flag14:/bin/bash
```

```
┌──(kali㉿kali)-[~]
└─$ gcc getflag.c
┌──(kali㉿kali)-[~]
└─$ ./a.out 'g <t61:|4_|!@IF.-62FH&G~DCK/Ekrvvdwz?v|'
flag is = 7QiHafiNa3HVozsaXkawuYrTstxbpABHD8CPnHJ
```