Challenge Description:

As a cybersecurity consultant, you've been tasked with assessing the security of a prominent corporation's network. Your mission is to penetrate their encrypted secret vault, housing valuable trade secrets and sensitive data, all safeguarded by a password system. Can you successfully breach the defenses and gain entry to the corporate vault.

Analysing the File :

```
┌──(kali㉿kali)-[~/Documents]
└─$ file sherlock
sherlock: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=fef76e38b5ff92ed0d08870ac523f9f3f8925a40, not stripped
```

ELF stands for Executable and Linkable Format. It is a common file format for executable files, object code, shared libraries, and core dumps. ELF files are used on Linux and other Unix-based systems.
The ELF format is versatile and can be executed on various processor types. It supports big-endian, little-endian, 32-bit, and 64-bit architectures systems and different CPUs.
The ELF format has several capabilities, including dynamic linking, dynamic loading, imposing run-time control on a program, and an improved method for creating shared libraries.The ELF format is the standard binary format on operating systems such as Linux.

```
┌──(kali㉿kali)-[~/Documents]
└─$ ./sherlock
Usage: ./sherlock password

┌──(kali㉿kali)-[~/Documents]
└─$ ./sherlock dadsfda
Access denied.
```

You need to enter the correct password to access the secret vault.

Strings:

```
┌──(kali㉿kali)-[~/Documents]
└─$ strings sherlock
/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
puts
printf
memset
atoi
__libc_start_main
/usr/local/lib:$ORIGIN
__gmon_start__
GLIBC_2.0
PTRh
=AVONt
j<jA
[^_]
UWVS
t$,U
[^_]
Usage: %s password
Access denied.
Access granted.
;*2$"(
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.9) 5.4.0 20160609
crtstuff.c
__JCR_LIST__
deregister_tm_clones
__do_global_dtors_aux
completed.7209
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
conditional2.c
giveFlag
__FRAME_END__
```

Looking into Main Function

Analysing in Ghidra

```
undefined4 main(int param_1,undefined4 *param_2)

{
  undefined4 uVar1;
  int iVar2;

  if (param_1 == 2) {
    iVar2 = atoi((char *)param_2[1]);
    if (iVar2 == 0x4e4f5641) {
      puts("Access granted.");
      giveFlag();
      uVar1 = 0;
    }
    else {
      puts("Access denied.");
      uVar1 = 1;
    }
  }
  else {
    printf("Usage: %s password\n",*param_2);
    uVar1 = 1;
  }
  return uVar1;
}
```

It checks if the program is invoked with exactly two arguments (`param_1 == 2`)
If the program is called with two arguments, it proceeds to convert the second argument (`param_2[1]`) to an integer using the `atoi` function.It checks if the converted integer matches the hexadecimal value `0x4e4f5641` which is equavalent to decimal value `1313822273` .If the password matches, it prints "Access granted.", calls the function `giveFlag()`, and returns `0` to indicate successful execution. If the password doesn't match, it prints "Access denied." and returns `1` to indicate failure.
From this we can find that we need to input the `1313822273` to get access to the vault and

lets look into `giveFlag()` function.

```
void giveFlag(void)

{
  int iVar1;
  undefined4 *puVar2;
  undefined4 *puVar3;
  char local_14c [60];
  undefined4 local_110 [60];
  uint local_20;

  puVar2 = &DAT_080486a0;
  puVar3 = local_110;
  for (iVar1 = 0x3c; iVar1 != 0; iVar1 = iVar1 + -1) {
    *puVar3 = *puVar2;
    puVar2 = puVar2 + 1;
    puVar3 = puVar3 + 1;
  }
  memset(local_14c,0x41,0x3c);
  for (local_20 = 0; local_20 < 0x3c; local_20 = local_20 + 1) {
    local_14c[local_20] = (char)local_110[local_20] + local_14c[local_20];
  }
  puts(local_14c);
  return;
}
```

```
{
  if ( argc == 2 )
  {
    if ( atoi(argv[1]) == 1313822273 )
    {
      puts("Access granted.");
      giveFlag();
      return 0;
    }
    else
    {
      puts("Access denied.");
      return 1;
    }
  }
```

```
  else
  {
    printf("Usage: %s password\n", *argv);
    return 1;
  }
}

//----- (08048524) ----------------------------------------------------------
int giveFlag()
{
  char s[60]; // [esp+0h] [ebp-148h] BYREF
  _BYTE v2[240]; // [esp+3Ch] [ebp-10Ch] BYREF
  unsigned int i; // [esp+12Ch] [ebp-1Ch]

  qmemcpy(v2, "\r", sizeof(v2));
  memset(s, 65, sizeof(s));
  for ( i = 0; i <= 0x3B; ++i )
    s[i] += v2[4 * i];
  return puts(s);
}
```

You can also solve this challenge using online decompiler like https://dogbolt.org/

dogbolt.org/?id=f780e6f5-0f2b-499e-9f23-e59a470c86c8

9.2.93

```
144         return;
145     }
146
147     int __libc_csu_fini()
148     {
149         unsigned int v1;  // eax
150
151         return v1;
152     }
153
154     int _fini()
155     {
156         return;
157     }
158
159
```

4.0.4911 (15e9149)

```
81
82
83  id __convention("reg
84
85    int32_t stack_end
86    __libc_start_main(
87    /* no return */
88
89
90  t32_t __x86.get_pc_t
91
92    return;
93
94
95  id deregister_tm_clo
96
97
```

Hex-Rays

8.3.0.230608

```
126
127 //----- (0804849B) ----------------------------------------------------------
128 int __cdecl main(int argc, const char **argv, const char **envp)
129 {
130   if ( argc == 2 )
131   {
132     if ( atoi(argv[1]) == 1313822273 )
133     {
134       puts("Access granted.");
135       giveFlag();
136       return 0;
137     }
138     else
139     {
140       puts("Access denied.");
141       return 1;
```

```
┌──(kali㉿kali)-[~/Documents]
└─$ ./sherlock 1313822273
Access granted.
NOVA{ReverseEng_expertise_unveiled_assemblyMaster_unlocked}
```

Flag

NOVA{ReverseEng_expertise_unveiled_assemblyMaster_unlocked}