Introduction

This CTF, "Feed The Magical Goat" is a Reverse Engineering challenge as part of the Battelle CTF challenges on https://www.battelle.org/the-challenge.

The CTF has the description of "*Once upon a time, there was a little reverse engineer who found a special bell. When the bell was struck, they say a magical billy goat appeared looking for food. Everyone knows billy goats will eat anything, but this is all the little reverse engineer had lying around.*"

Start

When you click the download button on the page you get a 'billygoat_executable.zip', after unzipping you receive the 'billygoat' file.

First I run the **file** bash command to see what type of file it is and see that it is a "ELF 32-bit LSB executable" so I know that I can open it in Ghidra.

First I try to just run the executable and I receive an introduction dialogue and an outro it seems.

```
┌─[kesifan@parrot]─[~/Documents/CTF/Battelle/billy]
└──╼ $./billygoat
You ring the chow bell...
OH NO, here comes Billygoat!!
Goats have microscopic 'hooks' on their hooves that give them unmatched
traction.
Actually no they don't, but that would be awesome!
Billygoat looks around... Angr fills his eyes.
Billygoat does a backflip and vanishes.
```

Main

After opening in Ghidra and looking in the main function I see the following C code which calls the give_offering with a string literal

```
  puts("You ring the chow bell...");
  sleep(2);
  puts("OH NO, here comes Billygoat!!");
  sleep(2);
  print_intro();
  sleep(2);
  __ptr = (undefined4 *)give_offering("chow.down");
```

give_offering

The following code from the give_offering function takes the string literal "chow.down" and attempts to open it as a file, giving an error message if it fails.

```
char * give_offering(char *param_1)

{
  int __fd;
  char *__buf;
  ssize_t sVar1;

  __fd = open(param_1,0);
  if (__fd == -1) {
    puts("Billygoat looks around... Angr fills his eyes.");
    close(-1);
    unlink(binName);
    print_outro();
                    /* WARNING: Subroutine does not return */
    exit(1);
  }
```

The following code then shows that it allocates a buffer of 16 bytes then attempts to read the first 16 bytes from "chow.down" and assign it to that buffer, then closes the file

```
  __buf = (char *)malloc(0x11);
  sVar1 = read(__fd,__buf,0x10);
  close(__fd);
```

There is more code but its not important, now we go back to main.

Buffer reading

Right after calling give_offering, the main function splits the 16 bytes into 4 local variables.

```
  local_34 = *__ptr;
  local_30 = __ptr[1];
  local_2c = __ptr[2];
  local_28 = __ptr[3];
```

The main function then calls 4 different functions, each with one of the variables.

```
  iVar1 = fill_rumen(&local_34);
  if (iVar1 != 0) {
    iVar1 = fill_reticulum(&local_30);
    if (iVar1 != 0) {
      iVar1 = fill_omasum(&local_2c);
      if (iVar1 != 0) {
        iVar1 = fill_abomasum(&local_28);
        if (iVar1 != 0) {
```

Each of these functions check each byte, so the first one, fill_rumen checks the first 4 bytes and checks to see if it matches the functions 4 bytes, for example.

```
  iVar1 = fill_rumen(&local_34);
  if (iVar1 != 0) {
    iVar1 = fill_reticulum(&local_30);
    if (iVar1 != 0) {
      iVar1 = fill_omasum(&local_2c);
      if (iVar1 != 0) {
        iVar1 = fill_abomasum(&local_28);
        if (iVar1 != 0) {
```

fill_rumen checks for ",", ";", ".", "u"
fill_reticulum checks for "P", "@", "h", "_"
fill_omasum checks for "|", "p", "n", "W"
fill_abomasum checks for "0x48", "B", "c", "r"

- After looking up an ASCII table, 0x48 is H

Reading the code further shows that if all this matches, you get the flag.

```
          puts(
              "Billygoat looks pleased. He bows to you. Congratulations, you
are now the goat master !\n"
              );
          printf("flag{%c%c%c%c_%c%c%c%c_%c%c%c%c_%c%c%c%c}\n",(int)
(char)local_34,
                 (int)local_34._1_1_,(int)local_34._2_1_,
(int)local_34._3_1_,(int)(char)local_30,
                 (int)local_30._1_1_,(int)local_30._2_1_,
(int)local_30._3_1_,(int)(char)local_2c,
                 (int)local_2c._1_1_,(int)local_2c._2_1_,
(int)local_2c._3_1_,(int)(char)local_28,
                 (int)local_28._1_1_,(int)local_28._2_1_,
(int)local_28._3_1_);
```

Conclusion

I put all of the correct characters in a file called "chow.down", ran the executable with the file and secured the flag

```
┌─[kesifan@parrot]─[~/Documents/CTF/Battelle/billy]
└──╼ $./billygoat chow.down
You ring the chow bell...
OH NO, here comes Billygoat!!
Goats have microscopic 'hooks' on their hooves that give them unmatched
traction.
Actually no they don't, but that would be awesome!
Billygoat eats your offering...
Billygoat looks pleased. He bows to you. Congratulations, you are now the
goat master!

flag{l1vn_th4t_goat_l1f3}
```