Introduction

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

This CTF has the description of *A mysterious figure has been terrorizing the village of Sleepy Hollow. He rides a massive horse, swings a might scythe and has been collecting heads from any who draw near. A group of locals, Ichabod Crane, Katrina Van Tassel and Abraham "Brom Bones" Van Brunt have been working to discover the secret behind this mysterious menace, but just as they were on the verge of putting the pieces together, the Headless Horseman struck!*

Start

After downloading the zip file, "headless_horseman.zip", and unzipping it, you get a directory called "body_bag". This contains

- /body_bag | directory
  - bloated_body | data file
  - decomposing_body | data file
  - rotting_body | data file
- headless_horseman | ELF 64-bit LSB pie executable
- README.txt

So firstly before I analyze headless_horseman in ghidra, I see whats in the README

```
┌─[kesifan@parrot]─[~/Documents/CTF/Battelle/horse/distributed_files]
└──╼ $cat README.txt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% The Legend of the Headless Horseman %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A mysterious figure has been terrorizing the village of Sleepy Hollow.

He rides a massive horse, swings a mighty scythe and has been collecting
heads from any who draw near.

A group of locals, Ichabod Crane, Katrina Van Tassel, and Abraham "Brom
Bones" Van Brunt have been working to discover the secret behind this
mysterious menace, but just as they were on the verge of putting the pieces
together, the Headless Horseman Struck!
```

```
All that are left of the heros are some unidentifiable bodies with no heads!

Can you help put our heros back together, and figure out what secrets they
uncovered? You'll first need to bargin with the horseman... bring some
pumpkins with you.. a LOT of pumpkins.
```

Running headless_horseman as an executable matches the readme, after an intro dialogue, I get asked for a mystery amount of pumpkins.

Main

Checking the main function even further matches this, it runs a function that most likely takes in the inputted amount of pumpkins and assigns it into a variable, then it counts them.

```
undefined8 main(void)

{
  undefined4 uVar1;

  print_intro();
  uVar1 = offer_pumpkins();
  count_offering(uVar1);
  return 0;
}
```

count_offering

Looking into this function shows that it checks the inputted amount of pumpkins through two different checks, first_count and second_count, if it passes both checks, you get whatever the bag of heads is. Maybe next part of the CTF?

```
  int iVar1;

  iVar1 = first_count(param_1);
  if (iVar1 != 0) {
    puts("You hear a grunt of approval but the mumbling continues");
    sleep(2);
    iVar1 = second_count(param_1);
    if (iVar1 != 0) {
      puts(
          "The figure turns to you and nods, pulling out his bag of heads,
dumping them on the groun d in front of you"
          );
```

```
        dump_heads(param_1);
        return 1;
```

first_count & second_count

first_count checks if the highest 0x10(16 bits) of the input is equal to 0xdead

```
bool first_count(uint param_1)
{
  return param_1 >> 0x10 == 0xdead;
}
```

second_count checks if the input is equal to 0xface

```
bool second_count(int param_1)
{
  return param_1 == 0xface;
}
```

So lets convert each to an integer. I learnt that too do this you multiply each hex digit represented as their decimal equivalent by (16^n), where n is their place. Do this for each hex digit then add them all together

```
0xdead
13 * 16^3 = 53248
14 * 16^2 = 3584
10 * 16^1 = 160
13 * 16^0 = 13

Total = 57005

-----------------------------------------------------------------------------
----
0xface
15 * 16^3 = 61440
10 * 16^2 = 2560
12 * 16^1 = 192
14 * 16^0 = 14

Total = 64206
```

Now the parameter in first_count uses the right shift operator by 16 bits, so after shifting the input 16 bits, it needs to be 57005.

This is confusing though since its impossible to have 1 unchanged parameter, equal 64206 and 57005(after being shifted 16 bits to the right), there is no pointer to change the variable, so instead I check the assembly code of second_count.

```
        0010135b 81 65 fc        AND        dword ptr [RBP + local_c],0xffff
                 ff ff 00 00
        00101362 c1 65 fc 10     SHL        dword ptr [RBP + local_c],0x10
```

After reading this you can see it takes the input, local_c, and does the **AND** bitwise operation with 0xffff with the input, so it will only take the lower 16 bits, and the next operation does the **SHL(shift left)** operation with 0x10(16 bits). So while not shown in the decompiled C code, this function takes the lower 16 bits of the given input.

So the first_count takes the high 16 bits, and the second_count take the low 16 bits. So if we made the input 0xDEADFACE in decimal, it should work

```
0xDEADFACE
13 * 16^7 = 3489660928
14 * 16^6 = 234881024
10 * 16^5 = 10485760
13 * 16^4 = 851968
15 * 16^3 = 61440
10 * 16^2 = 2560
12 * 16^1 = 192
14 * 16^0 = 14

Total = 3735943886
```

After running the program and putting that number as the input, he gives you a lot of heads, more than just the heads from the 3 bodies, and a hint to use **QEMU**

```
┌─[kesifan@parrot]─[~/Documents/CTF/Battelle/horse/distributed_files]
└──╼ $./headless_horseman
```

```
You see a dark figure looming in the darkness
As you approach he raises his hand to stop you
The figure holds up a bloody sack as if offering the contents to you
He holds out his other hand, as if expecting some kind of offering
you look back in your cart and scan over the pumpkins you brought.. will it
be enough?
how many pumpkins did you bring with you this time? 3735943886
The figure pulls a head off the saddle and holds it over the cart and you
```

```
hear it mumble as it inspects your offering
You hear a grunt of approval but the mumbling continues
The figure turns to you and nods, pulling out his bag of heads, dumping them
on the ground in front of you
A quick count indicates more heads than you were expecting, he is quite the
collector!
well, you seem have gotten what you came for..time to start stitching
As you pick up the first head you begin to wonder which body it might belong
to, and how on earth you might go about reviving these poor souls...
maybe you can use the fabled Quick and Efficient Murder Un-Doer(QEMU for
short)
```

rotting_body

I first started with rotting_body. I used "strings" on the data file and found this "/lib/ld-linux.so.2", which is the linker for i386 architecture systems.
After reading all of the "readelf -h" of the 6 heads, exactly one of them had the Intel 80386 machine architecture.
So then I combined them and got a full ELF file.

- cat ../shrunken_head rotting_body > maybe_full_elf

I then opened this in Ghidra, main led to the bron() function. This shows two variables, local_24 which is the input variable and local_10

```
  bool bVar1;
  undefined local_24 [20];
  int local_10;
  local_10 = -0x21524111
```

Farther down we see that its expected local_10

```
  FUN_000110c0(local_24);
  FUN_000110b0("Brom\'s eyes glaze over for a second and he writes down this
number: 0x%x\n",
               local_10);
  bVar1 = local_10 != 0x44414544;
  if (bVar1) {
    FUN_000110e0("Brom shakes himself off again");
    FUN_000110d0(1);
    FUN_000110e0(
                "\'Nope, that didn\'t seem to do it.. could you try again?
REALLY cram it down my th roat, I want to be overflowing with medicine!\'"
```

```
                );
  }
  else {
    FUN_000110e0("\'WOW! I think that did the trick, it\'s all coming back
to me now\'");
    FUN_000110d0(2);
    FUN_000110e0("\'here is my piece to this creepy puzzle, though I have no
idea what it means..\'"
                );
    FUN_000110d0(2);
    print_flag();
  }
```

After reading this and converting some little-endian hex, local_10 is equal to 0xdeadbeef, but the flag only prints if it equals 0xDEAD(it was shown in ghidra as 0xDAED because of big endian)

I ran the program with QEMU
So I overloaded the buffer of 20 bytes and put DEAD into there, and got what seems to be the end of the flag

```
┌─[kesifan@parrot]─[~/Documents/CTF/Battelle/horse/distributed_files/body_ba
g]
└──╼ $qemu-i386-static -L /usr/i386-linux-gnu ./maybe_full_elf
Brom shakes himself off as he stands up
'Well that was certainly an experience' he says, 'thanks for the help!'
You see him shake his head.. 'though i'm not sure you screwed me back
perfectly.. something feels a bit off'
'think you have any medicine to help straighten out my thoughts?'
aaaaaaaaaaaaaaaaaaaaDEAD
Brom's eyes glaze over for a second and he writes down this number:
0x44414544
'WOW! I think that did the trick, it's all coming back to me now'
'here is my piece to this creepy puzzle, though I have no idea what it
means..'
pumpkin_pie}
┌─[kesifan@parrot]─[~/Documents/CTF/Battelle/horse/distributed_files/body_ba
g]
└──╼ $
```

bloated_body

I used strings on bloated_body and saw that it had **MIPS** data within it and after reading the elf headers of the 6 heads, there is exactly 1 head with **MIPS** data. So I combined them with

- cat ../moldy_head bloated_body > mippy

After analyzing the file in ghidra, and checking the main function, it leads to the ichabod function()

```
  iVar1 = check_surroundings();
  if (iVar1 != 0) {
    puts("He appears to not find what he is looking for and collapses back
to the ground");
  }
  else {
    printf("\'ah, my trusty steed %s is here, all is well\' he says, pulling
something out of a sadd lebag\n"
           ,horse_name);
    sleep(1);
    puts("\'Here, I don\'t have the whole secret, but here is the piece I
was able to find\'");
    print_incantation();
  }
```

So now I need to check the check_surroundings() function to get the flag

```
{
  char *__s;
  __s = getenv("ICHABODS_HORSE");
  puts(__s);
  strcmp(__s,horse_name);
  return;
}
```

This shows that the C file checks for a environmental variable called "ICHABODS_HORSE" matching the global variable "horse_name", which I checked in Ghidra, equals GUNPOWDER

```
└─$ $ICHABODS_HORSE=GUNPOWDER qemu-mips-static mippy
Ichabod Crane gasps as life returns to his body
He begins looking around frantically
GUNPOWDER
'ah, my trusty steed GUNPOWDER is here, all is well' he says, pulling
something out of a saddlebag
'Here, I don't have the whole secret, but here is the piece I was able to
```

```
find'
flag{the_horseman_just_
```

decomposing_body

After using "strings" on this body, it shows that this is an ARM file, and there is exactly one head that is an **ARM ELF header**

- cat ../dessicated_head decomposing_body > decompose

This is great since I can run ARM on my local machine, no QEMU. So first i open this in Ghidra

```
  puts(
      "\'can you help me out? I was never very creative with these things,
maybe try the street I gr ew up on? or my Home Town?\'"
      );
  printf("What should Katrina use as the decryption key? ");
  fgets((char *)abStack_70,0x14,(FILE *)stdin);
  puts("\'You really think it\'s that?\'");
  sleep(1);
  puts("\'Well i\'ll give that a shot, does this look right?\'");
  for (local_74 = 0; local_74 < 0xe; local_74 = local_74 + 1) {
    abStack_70[local_74] = (&encrpyted_words)[local_74] ^
abStack_70[local_74];
  }
```

The program gives the hint that it could be the Home Town or street, and in the story the horseman terrorizes the town of "sleepy hollow", so I try that as the input and got the answer

```
└──╼ $./decompose
Katrina blinks awake, seeming a bit shocked ot be waking up again
'Oh! hello there! just before the lights went out I was working with Ichabod
and Brom to get rid of that pesky horseman for good!'
'Drat! it looks like I encrypted my portion but I cant seem to remember what
I used!'
'can you help me out? I was never very creative with these things, maybe try
the street I grew up on? or my Home Town?'
What should Katrina use as the decryption key? Sleepy Hollow
'You really think it's that?'
'Well i'll give that a shot, does this look right?'
really_loves_
```

Final

Putting all of the flag parts together I get
flag{the_horseman_just_really_loves_pumpkin_pie}