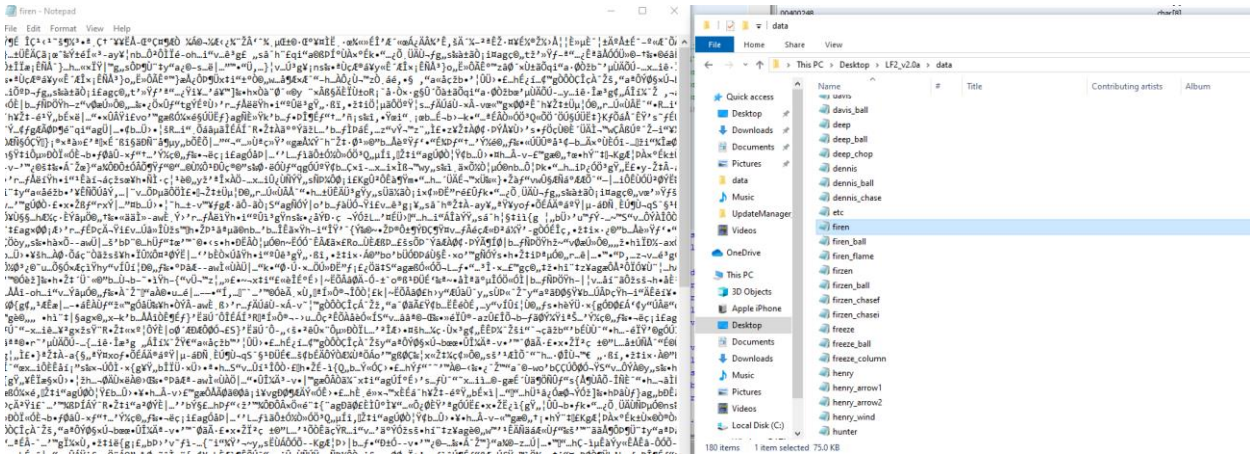
Reversing Little Fighter 2 for fun and profit

After downloading and installing the game we get a folder that includes executable, and game data folder.

Inspecting the data folder, we notice some interesting ".dat" files which their names matching the character's names in the game.



The file content seems obfuscated.

Lets, analyze the executable and reverse engineer the obfuscation algo.

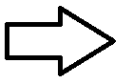
Opening the executable with "ghidra" and jumping right to the strings we see one filename reference which includes ".dat". Apparently, that filename is passed to a function, probably the de obfuscation function.

Ghidra gives us nice pseudo source code of the function. Lets analyze it.

```
void de_obfuscation(char *enc_file_ptr,char *dec_file_ptr)

{
  char cVar1;
  FILE *enc_file;
  undefined4 *key;
  int iVar2;
  uint uVar3;
  undefined4 *puVar4;
  int digit;
  FILE *dec_file;
  undefined local_30 [4];
  undefined4 local_2c [10];
  uint local_4;

  local_4 = DAT_0044eea4 ^ (uint)&digit;
  iVar2 = 9;
  key = (undefined4 *)"SiuHungIsAGoodBearBecauseHeIsVeryGood";
  puVar4 = local_2c;
  while (iVar2 != 0) {
    iVar2 = iVar2 + -1;
    *puVar4 = *key;
    key = key + 1;
    puVar4 = puVar4 + 1;
  }
  *(undefined2 *)puVar4 = *(undefined2 *)key;
  uVar3 = 0;
  enc_file = fopen(enc_file_ptr,"r");
  dec_file = fopen(dec_file_ptr,"w");
  digit = 0x7b;
```



Copies first 10 digits of key to local_2c var

First the function copies the first 10 digits of the hard-coded key to another variable, local_2c.

After that, we see that the function prepares 2 file IO's buffer's, one for reading and second for writing.

Then we enter 2 loops.

```
do {
  key = sub_key;
  do {
    cVar1 = *(char *)key;
    key = (undefined4 *)((int)key + 1);
  } while (cVar1 != '\0');
  uVar3 = (uVar3 + 1) % (uint)((int)key - ((int)sub_key + 1));
  fscanf(enc_file,"%c",local_30);
  counter = counter + -1;
} while (counter != 0);
fscanf(enc_file,"%c",&counter);
iVar2 = feof(enc_file);
while (iVar2 == 0) {
  counter = counter - *(char *)((int)sub_key + uVar3);
  if (counter < 0) {
    counter = counter + 0x100;
  }
  key = sub_key;
  do {
    cVar1 = *(char *)key;
    key = (undefined4 *)((int)key + 1);
  } while (cVar1 != '\0');
  uVar3 = (uVar3 + 1) % (uint)((int)key - ((int)sub_key + 1));
  fprintf(dec_file,"%c",counter);
  counter = 0;
  fscanf(enc_file,"%c",&counter);
  iVar2 = feof(enc_file);
}
fclose(enc_file);
fclose(dec_file);
FUN_004450b2();
return;
```

decrypted charcter

Lets focus on the second loop.

At the second loop we see that a digit is generated. Then , the "fprintf" function call write that digit to the decrypted file IO buffer.

The renamed variable "counter" acts as index at the enc_file buffer.

Each loop iteration the decrypted_char = enc_file[counter] – sub_key[index].

“uVar3” acts as the index variable.

According to the pseudo source code generated by ghidra uvar3 is defined as:

```
uVar3 = (uVar3 + 1) % (uint)((int)key - ((int)sub_key + 1));
```

(Key – sub_key+1) equals to the key size which is 0x25 (37 decimal).

Now we have to figure out what is starting value of uVar3.

```
28    uVar3 = 0;
29    enc_file = fopen(enc_file_ptr,"r");
30    dec_file = fopen(dec_file_ptr,"w");
31    counter = 0x7b;
32    do {
33      key = sub_key;
34      do {
35        cVar1 = *(char *)key;
36        key = (undefined4 *)((int)key + 1);
37      } while (cVar1 != '\0');
38      uVar3 = (uVar3 + 1) % (uint)((int)key - ((int)sub_key + 1));
39      fscanf(enc_file,"%c",local_30);
40      counter = counter + -1;
41    } while (counter != 0);
```

Looking at the first loop, uVar3 is initialized to 0, Also a counter variable is initialized to 0x7b (123 decimal).

Each iteration of the loop uVar3 is re assign as uVar3 = (uVar3 + 1) % (length(key))

There are 123 iterations so 123 % 37 = 12 = 0xc.

Now we can Decrypt the “.dat” file.

The encryption process is simply the inverse operation of the decryption function.

(Look at the included python script for decryption and encryption)

After decrypting “firzen.dat” we get clear text character properties file:

```
<bmp_begin>
name: Firzen
head: sprite\sys\firzen_f.bmp
small: sprite\sys\firzen_s.bmp
file(0-69): sprite\sys\firzen_0.bmp  w: 79  h: 79  row: 10  col: 7
file(70-139): sprite\sys\firzen_1.bmp  w: 79  h: 79  row: 10  col: 7
file(140-142): sprite\sys\firzen_2.bmp  w: 159  h: 79  row: 3  col: 1
walking_frame_rate 3
walking_speed 20.000000
walking_speedz 20.570000
running_frame_rate 3
running_speed 33.000000
running_speedz 5.670000
heavy_walking_speed 4.800000
heavy_walking_speedz 2.400000
heavy_running_speed 10.000000
heavy_running_speedz 1.200000
jump_height -17.000000
jump_distance 30.000000
jump_distancez 3.750000
dash_height -10.000000
dash_distance 22.000000
dash_distancez 5.000000
rowing_height -2.000000
rowing_distance 6.000000
<bmp_end>

<frame> 0 standing
   pic: 0  state: 0  wait: 10  next: 1  dvx: 0  dvy: 0  dvz: 0  centerx
   bpoint:
      x: 40  y: 34
   bpoint_end:
   wpoint:
      kind: 1  x: 17  y: 57  weaponact: 23  attacking: 0  cover: 0  dvx
   wpoint_end:
   bdy:
      kind: 0  x: 21  y: 18  w: 43  h: 62
   bdy_end:
<frame_end>

<frame> 1 standing
```

For the sake of demo I Changed the speed and jumping values.