Math

I opened the binary in IDA and saw this as the main function:

|  |
| --- |
| int \_\_cdecl main(int argc, const char \*\*argv, const char \*\*envp) {  int result; // eax  unsigned \_\_int8 v4; // [rsp+2h] [rbp-2Eh]  char v5; // [rsp+3h] [rbp-2Dh]  int v6; // [rsp+4h] [rbp-2Ch]  int v7; // [rsp+8h] [rbp-28h]  int i; // [rsp+Ch] [rbp-24h]  signed int j; // [rsp+10h] [rbp-20h]  signed int k; // [rsp+14h] [rbp-1Ch]  signed int l; // [rsp+18h] [rbp-18h]  int m; // [rsp+1Ch] [rbp-14h]   printf("Enter password: ", argv, envp);  scanf("%s", plaintext);  if ( strlen(plaintext) <= 0x100 )  {  v7 = 0;  v5 = 0;  for ( i = 0; i < strlen(plaintext); i += 3 )  {  v6 = key ^ (plaintext[i + 2] | ((plaintext[i + 1] | (plaintext[i] << 8)) << 8));  for ( j = 0; j <= 2; ++j )  {  if ( !plaintext[i + j] )  v5 = 1;  }  for ( k = 3; k >= 0; --k )  {  v4 = 0;  for ( l = 5; l >= 0; --l )  {  if ( v6 & (1 << (6 \* k + l)) )  v4 |= 1 << l;  }  if ( v4 )  {  ciphertext[v7] = base64[v4];  }  else if ( v5 )  {  ciphertext[v7] = 61;  }  else  {  ciphertext[v7] = 65;  }  ++v7;  }  }  for ( m = 0; flag[m]; ++m )  {  if ( flag[m] != ciphertext[m] )  {  puts(no);  return 0;  }  }  puts(yes);  result = 0;  }  else  {  puts("Error: password too long!");  result = 0;  }  return result; } |

A trained eye could easily deduce what the binary is doing: First it Exclusive-ORs the given plaintext with a hardcoded key (0x0DABEEF), then it encodes the string in base64 and finally compares the result with another hardcoded string (jveimeqpofewqY3chceAr+G6tPqKiM27u/CLhcbX7MPv). I used GDB to verify my suspicions, then just reversed the operations described ( decoded the base64 and Exclusive-ORed with 0x0DABEEF), and got the flag.