

Открываем файл в гидре, попробуем найти определенные дизассемблером строки. Однако ничего интересного мы там не находим, но программа определенно как-то запрашивает пароль и его сравнивает. Возможно, имеет место вызов функции сравнения. Попробуем найти strcmp

```
*****
*          POINTER to EXTERNAL FUNCTION          ***
*****
int __cdecl strcmp(char * _Str1, char * _Str2)
    int          EAX:4          <RETURN>
    char *       RCX:8          _Str1
    char *       RDX:8          _Str2
    0 strcmp <<not bound>>
PTR_strcmp_00405200                                XREF[1]:    strcmp:00404400
00405200 40 53 00      addr      MSVCRT.DLL::strcmp
        00 00 00
        00 00
```

Переходим по xref-функциям, чтобы найти место вызова strcmp

```
0040207e 4c 89 da      MOV     param_2,R11
00402081 e8 7a 23      CALL    MSVCRT.DLL::strcmp
        00 00
00402086 88 45 fb      MOV     byte ptr [RBP + local_d],AL
00402089 0f be 45      MOVSX   EAX,byte ptr [RBP + local_d]
```

Идем в дебаггер, ставим на этом адресе точку останова, запускаем программу и смотрим регистры

```
RAX  00000000014FE18  &"11$_zor0c%7^"
RBX  0000000000000000
RCX  0000000000971BC0  "password"
RDX  00000000009717E8  "11$_zor0c%7^"
RBP  00000000014FD90
RSP  00000000014FD60
RSI  0000000000000000
RDI  0000000000000000

R8   000000000000000A
R9   0000000000971658
R10  0000000000971BC0  "password"
R11  00000000009717E8  "11$_zor0c%7^"
R12  0000000000000000
R13  0000000000000000
R14  0000000000000000
R15  0000000000000000
```

Пароль найден. Проверяем:

```
Zoro Virtual Machine by vexor  
Enter password: 1l$_zor0c%7^  
Correct!
```

Вывод:

Не забывать про поиск типовых функций сравнения строк и других объектов\переменных. Тогда можно будет потратить меньше времени, чем если отсматривать весь дизассемблированный код и смотреть пошагово отладку.