

Решение Reverse-10 Редстоуновый механизм

Описание

В заброшенной крепости ты нашел сундук, защищенный сложным редстоуновым механизмом. Рядом с каждым рычагом стоит табличка со странными символами и числами. Изучи схему и найди правильную последовательность, чтобы активировать механизм и открыть сундук с алмазами!

Решение

Скачиваем исходные файлы, смотрим в основной файл с кодом на языке C - task.c. Видим что у нас обычный чекер пароля.

```
#include <stdio.h>
#include <string.h>

int main() {
    char password[32];
    printf("Enter password: ");
    fflush(stdout);

    if (scanf("%31s", password) != 1) {
        printf("No input\n");
        return 1;
    }

    if (strlen(password) != 8) { printf("Wrong!\n"); return 1; }
    if (password[0] != 'R') { printf("Wrong!\n"); return 1; }
    if ((password[1] ^ 0x1) != '3') { printf("Wrong!\n"); return 1; }
    if ((password[2] ^ 0x10) != 'f') { printf("Wrong!\n"); return 1; }
    if (password[3] + password[4] != 200) { printf("Wrong!\n"); return 1; }
    if (password[5] != 's') { printf("Wrong!\n"); return 1; }
    if ((password[6] - 'a') != 3) { printf("Wrong!\n"); return 1; }
    if (!(password[7] == '!' || password[7] == '1')) { printf("Wrong!\n");
return 1; }
    printf("Correct! Here is your flag: vsosh{n0_fl4g_h3r3}\n");
    return 0;
}
```

Рассмотрим все if'ы построчно

Разбор проверок (пошагово)

1. `strlen(password) == 8` — пароль ровно 8 символов.
 2. `password[0] == 'R'` — первый символ 'R'.
 3. `(password[1] ^ 0x1) == '3'` — значит `password[1] = '3' ^ 0x1`. В обратную сторону: `ord('3') = 0x33`; `0x33 ^ 0x01 = 0x32 = ord('2')`. Значит `password[1] = '2'`.
 4. `(password[2] ^ 0x10) == 'f'` — значит `password[2] = ord('f') ^ 0x10`. `ord('f') = 0x66`; `0x66 ^ 0x10 = 0x76 = ord('v')`. Значит `password[2] = 'v'`.
 5. `password[3] + password[4] == 200` — ищем пару ASCII, сумма 200. Удобный вариант: `101 ('e') + 99 ('c') = 200`. Значит `password[3] = 'e'`, `password[4] = 'c'`.
 6. `password[5] == 's'` — пятый индекс — 's'.
 7. `(password[6] - 'a') == 3` — `password[6] = 'a' + 3 = 'd'`.
 8. `password[7] == '!' || password[7] == '1'` — последний символ может быть '!' или '1'. Возьмём '1'.
- Собираем: `R 2 v e c s d 1` → `R2vecsd1` либо `R2vecsd!`.

Примерный скрипт-решение на языке Python

```
def recover_password():
    # 0-й символ
    c0 = 'R'
    # 1-й: (c1 ^ 0x1) == '3' => c1 = ord('3') ^ 0x1
    c1 = chr(ord('3') ^ 0x1)
    # 2-й: (c2 ^ 0x10) == 'f' => c2 = ord('f') ^ 0x10
    c2 = chr(ord('f') ^ 0x10)
    # 3-й и 4-й: sum == 200. Выберем ('e', 'c') как удобную пару
    c3 = 'e'
    c4 = 'c'
    assert ord(c3) + ord(c4) == 200
    # 5-й: 's'
    c5 = 's'
    # 6-й: (c6 - 'a') == 3 => c6 = chr(ord('a') + 3)
    c6 = chr(ord('a') + 3)
    # 7-й: '1' или '!'
    c7 = '1'
    pwd = ''.join([c0, c1, c2, c3, c4, c5, c6, c7])
    return pwd

if __name__ == '__main__':
```

```
password = recover_password()  
print('Recovered password:', password)
```

Запуск `python3 solve.py` выведет:

```
Recovered password: R2vecsd1
```

После получения пароля подключаемся к серверу и вводим пароль, получаем ответ в виде флага:

vsosh{r3dst0n3_0v3rd0s3}

Пояснение

- В проверке суммы (3+4) можно подобрать другие пары символов, дающие 200, но "e" + "с" — удобный и часто используемый вариант (101 + 99).