AlexCTF 2017

Catalyst Write Up

Author : pandakim2122

Date : 2017.12.05

이 파일을 실행하면 Username과 Password를 입력받도록 되어있습니다.

```
putchar(10);
printf("Username: ");
__isoc99_scanf("%s", username);
printf("Password: ", username);
__isoc99_scanf("%s", password);
printf("Logging in", password);
fflush(stdout);
```

입력받은 Username의 길이를 아래와 같이 검증합니다.

```
__int64 __fastcall sub_400C41(int len)
{
  __int64 result; // rax

  if ( 4 * (len >> 2) != len
    || 4 * (len >> 4) == len >> 2
    || (result = (unsigned int)(len >> 3), !(_DWORD)result)
    || len >> 4 )
  {
    puts("invalid username or password");
    exit(0);
  }
  return result;
}
```

이 조건을 정리하면, "4의 배수이며, 8~15 사이를 만족하는 username의 길이는 8 또는 12이다." 가 됩니다.

조건	설명
4 * (len >> 2) != len	len이 4의 배수가 아니다
4 * (len >> 4) == len >> 2	len을 16으로 나누고, 4를 곱한 값이 len을 4로 나눈 값과 같다. 즉, len이 4 이상이다.
result = (unsigned int)(len >> 3), !(_DWORD)result	len을 8로 나눈 몫이 없다. 즉, len이 8 미만이다.
len >> 4	len을 16으로 나누었을 때 몫이 존재한다. 즉, len이 16 이상이다.

다음 함수에서 username 을 4 바이트씩 끊어서 읽습니다. exit(0)로 빠지지 않기 위한 조건을 만족하기 위해 3 개의 연산을 방정식으로 변환하면 다음과 같습니다.

A – B + C = 0x5c664b56

B + 3 * (C + A) = 0x2e700c7b2

C * B = 0x32ac30689a6ad314

이 방정식을 풀면 각각의 값은 다음과 같습니다.

A = 0x61746163

B = 0x7473796C

C = 0x6F65635F

이 값을 아스키로 변환하면, catalyst_ceo가 됩니다.

Username : catalyst_ceo

```
signed __int64 __fastcall sub_400CDD(unsigned int *username)
{
  signed __int64 result; // rax
  __int64 int_8; // [rsp+10h] [rbp-20h]
  __int64 int_4; // [rsp+18h] [rbp-18h]
  __int64 int_0; // [rsp+20h] [rbp-10h]

  int_0 = *username;
  int_4 = username[1];
  int_8 = username[2];
  if ( int_0 - int_4 + int_8 != 0x5C664B56
    || int_4 + 3 * (int_8 + int_0) != 0x2E700C7B2LL
    || (result = 0x32AC30689A6AD314LL, int_8 * int_4 != 0x32AC30689A6AD314LL) )
  {
    puts("invalid username or password");
    exit(0);
  }
  return result;
}
```

패스워드 검증 루틴인 이 곳에선 username을 4바이트씩 끊어, 더한 값(0x454D3E2E)을 srand()의 seed로 지정합니다. 이후 password를 4바이트씩 끊어 rand()와 sub 연산을 한 후, 4바이트의 값과 비교합니다.

```
srand(username[1] + *username + username[2]);
v2 = *password;
if ( v2 - rand() != 0x55EB052A )
{
  puts("invalid username or password");
  exit(0);
}
v3 = password[1];
if ( v3 - rand() != 0xEF76C39 )
{
  puts("invalid username or password");
  exit(0);
}
```

다음과 같은 코드로 exit(0)으로 빠지지 않는 password의 값을 구할 수 있습니다.

```
int main()
{
    srand(0x454D3E2E);
    unsigned int a[10] = {0x55EB052A, 0xEF76C39, 0xCC1E2D64, 0xC7B6C6F5, 0x26941BFA, 0x260CF0F3, 0x10D4

    for( int i=0; i<10; i++){
        unsigned int r = rand();
        printf("rand: %x, data: %x, result: %x, ", r, a[i], r+a[i]);
        printf("result_ascii: %c%c%c%c\r\n", r+a[i], (r+a[i])>>8, (r+a[i])>>16, (r+a[i])>>24);
    }

    return 0;
}
```

input

```
rand: 684749, data: 55eb052a, result: 56534c73, result_ascii: sLSV
rand: 673ce537, data: ef76c39, result: 76345170, result_ascii: pQ4v
rand: 7b4505e7, data: cc1e2d64, result: 4763334b, result_ascii: K3cG
rand: 70a0b262, data: c7b6c6f5, result: 38577957, result_ascii: WyW8
rand: 33d5253c, data: 26941bfa, result: 5a694136, result_ascii: 6AiZ
rand: 515a7675, data: 260cf0f3, result: 77676768, result_ascii: hggw
rand: 596d7d5d, data: 10d4caef, result: 6a42484c, result_ascii: LHBj
rand: 7cd29049, data: c666e824, result: 4339786d, result_ascii: mx9C
rand: 59e72db6, data: fc89459c, result: 56707352, result_ascii: RspV
rand: 4654600d, data: 2413073a, result: 6a676747, result_ascii: Gggj
```

Password : sLSVpQ4vK3cGWyW86AiZhggwLHBjmx9CRspVGggj

마지막으로, 입력받은 패스워드로 아래의 값과 Byte XOR 을 통해 최종 결과를 출력합니다.

- 0x42, 0x13, 0x27, 0x62, 0x41, 0x35, 0x6B, 0x0F, 0x7B, 0x46, 0x3C, 0x3E, 0x67, 0x0C, 0x8, 0x59, 0x44, 0x72, 0x36, 0x5, 0x0F, 0x15, 0x54, 0x43, 0x38, 0x17, 0x1D, 0x18, 0x8, 0x0E, 0x5C, 0x31, 0x21, 0x16, 0x2, 0x9, 0x18, 0x14, 0x54, 0x59

```
printf("your flag is: ALEXCTF{", a2, a1);
for ( i = 0; i < strlen(s); ++i )
  putchar((unsigned __int8)byte_6020A0[i] ^ s[i]);
return puts("}");
}
```

your flag is: ALEXCTF{1_t41d_y0u_y0u_ar3__gr34t__reverser__s33}