Ez Crackme writeup

Tools used : ghidra for code analyzing ,gdb for viewieing assembly

```
local_24 = strcmp(local_3c,">XWWabw$}-.");
if (local_24 == 0) {
  printf("correct password!!!");
```

pwndbg> disass secret
Dump of assembler code for function secret:
0x0000000000001159 <+0>: push rbp
0x000000000000115a <+1>: mov rbp,rsp
0x000000000000115d <+4>: mov eax,edi
0x000000000000115f <+6>: mov BYTE PTR [rbp-0x14],al
0x0000000000001162 <+9>: mov edx,DWORD PTR [rip+0x2ec4] #
0x402c <counter.0>
0x0000000000001168 <+15>: mov eax,edx
0x000000000000116a <+17>: shl eax,0x4
0x000000000000116d <+20>: sub eax,edx
0x000000000000116f <+22>: add eax,0x38
0x0000000000001172 <+25>: mov DWORD PTR [rbp-0x4],eax
0x0000000000001175 <+28>: movsx edx,BYTE PTR [rbp-0x14]
0x0000000000001179 <+32>: mov eax,DWORD PTR [rbp-0x4]
0x000000000000117c <+35>: mov ecx,eax
0x000000000000117e <+37>: shr ecx,0x1f
0x0000000000001181 <+40>: add eax,ecx
0x0000000000001183 <+42>: sar eax,1
0x0000000000001185 <+44>: add edx,eax
0x0000000000001187 <+46>: movsxd rax,edx
0x000000000000118a <+49>: imul rax,rax,0x2aaaaaab
0x0000000000001191 <+56>: shr rax,0x20
0x0000000000001195 <+60>: mov ecx,eax

```
0x0000000000001197 <+62>:    sar    ecx,0x4
0x000000000000119a <+65>:    mov    eax,edx
0x000000000000119c <+67>:    sar    eax,0x1f
0x000000000000119f <+70>:    sub    ecx,eax
0x00000000000011a1 <+72>:    mov    eax,ecx
0x00000000000011a3 <+74>:    add    eax,eax
0x00000000000011a5 <+76>:    add    eax,ecx
0x00000000000011a7 <+78>:    shl    eax,0x5
0x00000000000011aa <+81>:    sub    edx,eax
0x00000000000011ac <+83>:    mov    ecx,edx
0x00000000000011ae <+85>:    mov    eax,ecx
0x00000000000011b0 <+87>:    add    eax,0x20
0x00000000000011b3 <+90>:    mov    BYTE PTR [rbp-0x5],al
0x00000000000011b6 <+93>:    mov    eax,DWORD PTR [rip+0x2e70]        #
0x402c <counter.0>
0x00000000000011bc <+99>:    add    eax,0x1
0x00000000000011bf <+102>:   mov    DWORD PTR [rip+0x2e67],eax        #
0x402c <counter.0>
0x00000000000011c5 <+108>:   movzx  eax,BYTE PTR [rbp-0x5]
0x00000000000011c9 <+112>:   pop    rbp
0x00000000000011ca <+113
```

What `secret()` does

From the assembly we can read that for each input byte `p` and a running `counter` (starting at 0), the function computes:

- `k = (counter * 15 + 0x38) // 2`
- `i = p + k`
- returns `((i % 96) + 32)` — i.e. reduce to the 96-printable range and add 0x20.

So the equation for the target character `T` at position `counter` is:

$$T = ((p + k) \% 96) + 32$$
$$\Rightarrow p \equiv (\text{ord}(T) - 32) - k \pmod{96}$$

So the password is bulganteng