REVERSE ENGINEERING

Author: Kyou

Rolling My Own - picoCTF 2021

Difficulty: Hard

Description:

I don't trust password checkers made by other people, so I wrote my own. It doesn't even need to store the password! If you can crack it I'll give you a flag.

Hint 1: It's based on this paper

https://link.springer.com/article/10.1007/s11416-006-0011-3

Hint 2: Here's the start of the password: D1v1

Author: Luke Rindels

Remote: nc mercury.picoctf.net 35226

SOLUTION

Pada soal ini diberikan sebuah binary ELF

```
remote: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=
f27ac845639bfc869ef1ccd69c5f80749bce5ecb, stripped
```

yang dimana setelah di decompile pada fungsi **main** akan menghasilkan potongan kode sebagai berikut.

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  unsigned int v3; // eax
  __int64 v4; // rdx
  int i; // [rsp+8h] [rbp-F8h]
  int j; // [rsp+8h] [rbp-F8h]
  int k; // [rsp+Ch] [rbp-F4h]
  void (__fastcall *shell)(unsigned __int64 (__fastcall *)(__int64)); // [rsp+10h] [rbp-F0h]
  _BYTE *ptr; // [rsp+18h] [rbp-E8h]
```

```
_DWORD pos[4]; // [rsp+20h] [rbp-E0h]
_QWORD v12[2]; // [rsp+30h] [rbp-D0h]
char v13[48]; // [rsp+40h] [rbp-C0h] BYREF
char s[64]; // [rsp+70h] [rbp-90h] BYREF
char dest[72]; // [rsp+B0h] [rbp-50h] BYREF
unsigned __int64 v16; // [rsp+F8h] [rbp-8h]

v16 = __readfsqword(0x28u);
setbuf(stdout, 0);
strcpy(v13, "GpLaMjEWpVOjnnmkRGiledp6Mvcezxls");
pos[0] = 8;
pos[1] = 2;
pos[2] = 7;
pos[3] = 1;
memset(s, 0, sizeof(s));
memset(dest, 0, 0x40u);
printf("Password: ");
fgets(s, 64, stdin);
s[strlen(s) - 1] = 0;
for ( i = 0; i <= 3; ++i )
{
  strncat(dest, &s[4 * i], 4u);
  strncat(dest, &v13[8 * i], 8u);
}
ptr = malloc(0x40u);
v3 = strlen(dest);
```

```
  hash(ptr, dest, v3);
  for ( j = 0; j <= 3; ++j )
  {
    for ( k = 0; k <= 3; ++k )
      *((_BYTE *)v12 + 4 * k + j) = ptr[16 * k + j + pos[k]];
  }
  shell = (void (__fastcall *)(unsigned __int64 (__fastcall *)(__int64)))mmap(0,
0x10u, 7, 34, -1, 0);
  v4 = v12[1];
  *(_QWORD *)shell = v12[0];
  *((_QWORD *)shell + 1) = v4;
  shell(flag);
  free(ptr);
  return 0;
}
```

Pada kodenya, password akan diproses dalam potongan 4 karakter `s[4*i..4*i+3]` yang dimana untuk setiap bloknya akan ditambahkan 8 char dari string GpLaMjEWpVOjnnmkRGiledp6Mvcezxls dan untuk tiap iterasinya ngehasilin 12 byte dari password dan stringnya (setelah 4 iterasi -> buffer totalnya akan sepanjang 48 byte).

Buffer bakalan diproses sama fungsi `hash` yang memakai MD5 per 12 byte block dan hasilnya akan ditaruh ke output buffer 64 byte dengan skema cyclic.

```
unsigned __int64 __fastcall hash(__int64 a1, __int64 a2, int len)
{
  int v3; // eax
  int i; // [rsp+20h] [rbp-90h]
  int j; // [rsp+24h] [rbp-8Ch]
  int v9; // [rsp+28h] [rbp-88h]
```

```
int v10; // [rsp+2Ch] [rbp-84h]
_BYTE v11[96]; // [rsp+30h] [rbp-80h] BYREF
_BYTE v12[24]; // [rsp+90h] [rbp-20h] BYREF
unsigned __int64 v13; // [rsp+A8h] [rbp-8h]

v13 = __readfsqword(0x28u);
if ( len % 12 )
  v3 = len / 12 + 1;
else
  v3 = len / 12;
v10 = v3;
for ( i = 0; i < v10; ++i )
{
  v9 = 12;
  if ( i == v10 - 1 && len % 12 )
    v9 = v10 % 12;
  MD5_Init(v11);
  MD5_Update(v11, a2, v9);
  a2 += v9;
  MD5_Final(v12, v11);
  for ( j = 0; j <= 15; ++j )
    *(_BYTE *)(a1 + (j + 16 * i) % 64) = v12[j];
}
return __readfsqword(0x28u) ^ v13;
}
```

Setelah itu, beberapa byte dari posisi output tertentu bakalan diextract yang akan membentuk array shellcode yang ukurannya 16 byte, yang setelahnya shellcode bakalan dipanggil dengan memberikan address dari fungsi `flag` sebagai sebuah argumen di rdi.

```
unsigned __int64 __fastcall flag(__int64 a1)
{
  FILE *stream; // [rsp+18h] [rbp-98h]
  char s[136]; // [rsp+20h] [rbp-90h] BYREF
  unsigned __int64 v4; // [rsp+A8h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  if ( a1 == 0x7B3DC26F1LL )
  {
    stream = fopen("flag", "r");
    if ( !stream )
    {
      puts("Flag file not found. Contact an admin.");
      exit(1);
    }
    fgets(s, 128, stream);
    puts(s);
  }
  else
  {
    puts("Hmmmmmm... not quite");
  }
  return __readfsqword(0x28u) ^ v4;
```

```
}
```

Jadinya kita harus bikin shellcode yang nge-call fungsi di rdi dengan mengirim 0x7B3DC26F1 ke rdi saat memanggil fungsi itu.

```
PS S:\revmaxxing\picoCTF\rolling_my_own> python3
>>> from pwn import asm
[*] You have the latest version of Pwntools (4.14.1)
>>> shellcode = """
... mov rsi, rdi
... mov rdi, 0x7B3DC26F1
... call rsi
... """
>>> asm(shellcode, arch="amd64").hex()
'4889fe48bff126dcb307000000ffd690'
```

Dan bagian terakhirnya kita tinggal bruteforce passwordnya hehe

Full solver:

```python
import hashlib

bytes = ["4889fe48", "bff126dc", "b3070000", "00ffd6"]

offsets = [8, 2, 7, 1]

strings = [

    "GpLaMjEW", "pVOjnnmk", "RGiledp6", "Mvcezxls"

]



def bf(block_idx: int) -> str:

    target = bytes[block_idx]

    offset = offsets[block_idx]

    salt = strings[block_idx]



    for a in range(33, 123):

        for b in range(33, 123):
```

```
            for c in range(33, 123):
                for d in range(33, 123):
                    cand = chr(a) + chr(b) + chr(c) + chr(d) + salt
                    md5_hex = hashlib.md5(cand.encode()).hexdigest()
                    if md5_hex[offset*2:offset*2+len(target)] == target:
                        return cand[:4]
    return None



if __name__ == "__main__":
    password = []

    for i in range(4):
        part = bf(i)
        if part:
            print(f"[+] Found block {i}: {part}")
            password.append(part)
        else:
            print(f"[-] Block {i} not found :(")

    full_password = "".join(password)
    print("\n[+] Final Password:", full_password)
[Running] python -u "s:\revmaxxing\picoCTF\rolling_my_own\solver.py"
[+] Found block 0: D1v1
[+] Found block 1: d3An
```

```
[+] Found block 2: dC0n

[+] Found block 3: \rpB

[+] Final Password: D1v1d3AndC0n\rpB

(base)ordinarycat@LAPTOP-0RRJ88VS:/mnt/s/revmaxxing/picoCTF/rolling_my_own
$ nc mercury.picoctf.net 35226

Password: D1v1d3AndC0n\rpB

picoCTF{r011ing_y0ur_0wn_crypt0_15_h4rd!_dae85416}

timeout: the monitored command dumped core
```

“Akhirnya solp juga ;_;”

-Kyou