PicoCTF 2018 - Keygen-me-2 writeup

Lê Quốc Dũng

April 2019

Solve: The software has been updated. Can you find us a new product key for the program in /problems/keygen-me-2_2_5374d282dd09441ebd3055cb968eefff

Hint: z3

Solution:

Sau khi decompile chương trình mình thấy được cấu trúc như sau:

```
undefined4 main(int param_1,int param_2)

{
  char cVar1;
  undefined4 uVar2;
  undefined4 *puVar3;

  puVar3 = &param_1;
  setvbuf(stdout,(char *)0x0,2,0);
  if (param_1 < 2) {
    puts("Usage: ./activate <PRODUCT_KEY>");
    uVar2 = 0xffffffff;
  }
  else {
    cVar1 = check_valid_key(*(undefined4 *)(param_2 + 4));
    if (cVar1 == 0) {
      puts("Please Provide a VALID 16 byte Product Key.");
      uVar2 = 0xffffffff;
    }
    else {
      cVar1 = validate_key(*(undefined4 *)(param_2 + 4));
      if (cVar1 == 0) {
        puts("INVALID Product Key.");
        uVar2 = 0xffffffff;
      }
      else {
        printf("Product Activated Successfully: ");
        print_flag(puVar3);
        uVar2 = 0;
      }
    }
  }
  return uVar2;
}
```

Mình cần truyền thêm 1 tham số là product key và mình cần kiểm tra 2 hàm check_valid_key và hàm validate_key

```
int check_valid_key(char *param_1)

{
  char cVar1;
  char local_9;
  int local_8;

  if (param_1 == (char *)0x0) {
    local_8 = 0;
  }
  else {
    local_9 = *param_1;
    local_8 = 0;
    while (local_9 != 0) {
      cVar1 = check_valid_char((int)local_9);
      if (cVar1 == 0) {
        return 0;
      }
      local_8 = local_8 + 1;
      local_9 = param_1[local_8];
    }
    if (local_8 != 0x10) {
      local_8 = 0;
    }
  }
  return local_8;
```

```
undefined4 validate_key(char *param_1)

{
  char cVar1;
  size_t sVar2;

  sVar2 = strlen(param_1);
  cVar1 = key_constraint_01(param_1,sVar2);
  if (((((cVar1 != 0) && (cVar1 = key_constraint_02(param_1,sVar2), cVar1 != 0)) &&
       (cVar1 = key_constraint_03(param_1,sVar2), cVar1 != 0)) &&
      ((((cVar1 = key_constraint_04(param_1,sVar2), cVar1 != 0 &&
         (cVar1 = key_constraint_05(param_1,sVar2), cVar1 != 0)) &&
        ((cVar1 = key_constraint_06(param_1,sVar2), cVar1 != 0 &&
          ((cVar1 = key_constraint_07(param_1,sVar2), cVar1 != 0 &&
            (cVar1 = key_constraint_08(param_1,sVar2), cVar1 != 0)))))) &&
       (cVar1 = key_constraint_09(param_1,sVar2), cVar1 != 0)))) &&
     (((cVar1 = key_constraint_10(param_1,sVar2), cVar1 != 0 &&
       (cVar1 = key_constraint_11(param_1,sVar2), cVar1 != 0)) &&
      (cVar1 = key_constraint_12(param_1,sVar2), cVar1 != 0)))) {
    return 1;
  }
  return 0;
}
```

Trong hàm check_valid_key lại gọi thêm hàm check_valid_char

```
undefined4 check_valid_char(char param_1)

{
  undefined4 uVar1;

  if (((param_1 < '0') || ('9' < param_1)) && ((param_1 < 'A' || ('Z' < param_1)))) {
    uVar1 = 0;
  }
  else {
    uVar1 = 1;
  }
  return uVar1;
}
```

Vậy là hàm check_valid_char sẽ kiểm tra 1 char có phải ký tự in hoa hoặc chữ số hay không. Nghĩa là product key của mình sẽ là 1 chuỗi 16 ký tự in hoa hoặc chữ số.

Quay trở lại hàm validate_key, mình thấy hàm này gọi 12 hàm con để kiểm tra tính hợp lệ của product key nên mình sẽ coi từng hàm để tìm cách sinh product key hợp lệ.

```
uint key_constraint_01(char *param_1)

{
  char cVar1;
  char cVar2;
  uint uVar3;

  cVar1 = ord((int)*param_1);
  cVar2 = ord((int)param_1[1]);
  uVar3 = mod((int)cVar2 + (int)cVar1,0x24);
  return uVar3 & 0xffffff00 | (uint)(uVar3 == 0xe);
}
```

```
uint key_constraint_02(int param_1)

{
  char cVar1;
  char cVar2;
  uint uVar3;

  cVar1 = ord((int)*(char *)(param_1 + 2));
  cVar2 = ord((int)*(char *)(param_1 + 3));
  uVar3 = mod((int)cVar2 + (int)cVar1,0x24);
  return uVar3 & 0xffffff00 | (uint)(uVar3 == 0x18);
}
```

```
uint key_constraint_11(int param_1)

{
  char cVar1;
  char cVar2;
  char cVar3;
  uint uVar4;

  cVar1 = ord((int)*(char *)(param_1 + 8));
  cVar2 = ord((int)*(char *)(param_1 + 9));
  cVar3 = ord((int)*(char *)(param_1 + 10));
  uVar4 = mod((int)cVar3 + (int)cVar1 + (int)cVar2,0x24)
  return uVar4 & 0xffffff00 | (uint)(uVar4 == 0x1b);
}

uint key_constraint_12(int param_1)

{
  char cVar1;
  char cVar2;
  char cVar3;
  uint uVar4;

  cVar1 = ord((int)*(char *)(param_1 + 7));
  cVar2 = ord((int)*(char *)(param_1 + 0xc));
  cVar3 = ord((int)*(char *)(param_1 + 0xd));
  uVar4 = mod((int)cVar3 + (int)cVar1 + (int)cVar2,0x24);
  return uVar4 & 0xffffff00 | (uint)(uVar4 == 0x17);
}
```

Hàm này lại gọi thêm hàm ord và hàm mod @@@.

```
int ord(byte param_1)

{
  int iVar1;

  if (((char)param_1 < '0') || ('9' < (char)param_1)) {
    if (((char)param_1 < 'A') || ('Z' < (char)param_1)) {
      puts("Found Invalid Character!");
                    /* WARNING: Subroutine does not return */
      exit(0);
    }
    iVar1 = (uint)param_1 - 0x37;
  }
  else {
    iVar1 = (uint)param_1 - 0x30;
  }
  return iVar1;
}

int mod(int param_1,int param_2)

{
  param_1 = param_1 % param_2;
  if (param_1 < 0) {
    param_1 = param_2 + param_1;
  }
  return param_1;
}
```

Hàm ord sẽ lấy mã ascii trừ 0x37 = 55 nếu là ký tự in hoa, lấy mã ascii trừ 0x30 = 48 nếu là chữ số. Nghĩa là hàm sẽ trả về giá trị từ 0 tới 35.

Hàm mod truyền vào 2 tham số a, b và trả về a mod b.

Vì product key của mình gồm 16 ký tự nên mình sẽ đặt ord của 16 ký tự đó là s[0], s[1], ..., s[15]. Nhìn vào hàm key_constraint_01 ở trên thì thấy hàm này kiểm tra điều kiện (s[0] + s[1]) mod 36 == 14. Còn hàm key_constraint_02 kiểm tra (s[2] + s[3]) mod 36 == 24. Và cứ như vậy, mình kiểm tra điều kiện ở 12 hàm: hàm key_constraint_11 kiểm tra (s[8] + s[9] + s[10]) mod 36 == 27 và hàm key_constraint_12 kiểm tra (s[7] + s[12] + s[13]) mod 36 == 23.

Việc tiếp theo chính là ..... brute force =)))) Các bạn có thể code 12 vòng lặp như mình =)))) hoặc dùng z3 theo như hint.

Trên thực tế có rất nhiều product key thỏa mãn. Nên mình lấy cái nào cũng được.

```
68CCQKB9Z7L38615
68CCQKB9Z7L39534
68CCQKB9Z7L48615
68CCQKB9Z7L49534
68CCQKB9Z7L58615
68CCQKB9Z7L59534
68CCQKB9Z7L68615
68CCQKB9Z7L69534
68CCQKB9Z7L78615
68CCQKB9Z7L79534
68CCQKB9Z7L88615
68CCQKB9Z7L89534
68CCQKB9Z7L98615
68CCQKB9Z7L99534
77DB
86EA
95F9
dung@DotDot:~$ ./Downloads/activate1 68CCQKB9Z7L99534
Product Activated Successfully: Flag File is Missing.
dung@DotDot:~$ ./Downloads/activate1 68CCQKB9Z7L98615
Product Activated Successfully: Flag File is Missing.
dung@DotDot:~$ ./Downloads/activate1 68CCQKB9Z7L68615
Product Activated Successfully: Flag File is Missing.
dung@DotDot:~$
```

Có product key rồi thì remote lên shell của pico và lấy flag thôi =)))))

Flag: $picoCTF\{c0n5tr41nt\_50lv1nG\_15_W4y\_f45t3r\_2923966318\}$

Chúc các bạn vui vẻ!!!