igeMoZI?

> UPX 패킹된 프로그램이 주어진다.

Analysis

```
cpstd@krrr:/mnt/c/Users/cpstd/Downloads/igeMOZI$ strings -tx ./igeMOZI.upx | grep
UPX
     eb VUPX!
  4ed0b $Info: This file is packed with the UPX executable packer
http://upx.sf.net $
  4ed5a $Id: UPX 3.96 Copyright (C) 1996-2020 the UPX Team. All Rights Reserved. $
  4f083 UPX!u
```

UPX 3.96 버전으로 패킹되었지만, 상용 프로그램으로 언패킹이 되지 않는다. 즉, `anti-upx`가 걸려있다. 그 방법으로는 UPX의 패킹 과정에서 생겨나는 UPX 헤더들을 망가뜨리는 방법이 있다. 이를 고쳐주는게 정석인 것 같지만, 그냥 메모리에서 덤프를 뜨기로 했다.

strace의 결과이다.

```
execve("./igeMOZI.upx", ["./igeMOZI.upx"], 0x7fffffffe2f0 /* 21 vars */) = 0
open("/proc/self/exe", O_RDONLY)        = 3
mmap(NULL, 325286, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ffff7faa000
mmap(0x7ffff7faa000, 324888, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0) =
0x7ffff7faa000
mprotect(0x7ffff7ff8000, 5798, PROT_READ|PROT_EXEC) = 0
readlink("/proc/self/exe", "/mnt/c/Users/cpstd/Downloads/ige"..., 4095) = 48
mmap(0x400000, 839680, PROT_NONE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x400000
mmap(0x400000, 1320, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x400000
mprotect(0x400000, 1320, PROT_READ)     = 0
mmap(0x401000, 615965, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0x1000) = 0x401000
mprotect(0x401000, 615965, PROT_READ|PROT_EXEC) = 0
mmap(0x498000, 165062, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0x98000) = 0x498000
mprotect(0x498000, 165062, PROT_READ)   = 0
mmap(0x4c1000, 25232, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0xc0000) = 0x4c1000
mprotect(0x4c1000, 25232, PROT_READ|PROT_WRITE) = 0
mmap(0x4c8000, 19520, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x4c8000
mmap(NULL, 4096, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ffff7fa9000
```

```
close(3)                                = 0
munmap(0x7ffff7faa000, 325286)          = 0
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffffffe240) = -1 EINVAL (Invalid argument)
brk(NULL)                               = 0x4cd000
brk(0x4cddc0)                           = 0x4cddc0
arch_prctl(ARCH_SET_FS, 0x4cd3c0)       = 0
set_tid_address(0x4cd690)               = 5146
set_robust_list(0x4cd6a0, 24)           = 0
rseq(0x4cdd60, 0x20, 0, 0x53053053)     = 0
uname({sysname="Linux", nodename="krrr", ...}) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0
readlink("/proc/self/exe", "/mnt/c/Users/cpstd/Downloads/ige"..., 4096) = 48
getrandom("\xdd\xeb\x8c\x37\xa2\x7d\xa9\x4a", 8, GRND_NONBLOCK) = 8
brk(0x4eedc0)                           = 0x4eedc0
brk(0x4ef000)                           = 0x4ef000
mprotect(0x4c1000, 16384, PROT_READ)    = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0
write(1, "flag : 89 \n", 11flag : 89
)            = 11
exit_group(0)                           = ?
+++ exited with 0 +++
```

`mmap`과 `mprotect` 함수를 다수 진행하고 있는데, 마지막 `munmap` 이후로는 정상적인 실행 함수들이 존재하는 것을 볼 수 있다.

```
gdb-peda$ catch syscall munmap
Catchpoint 1 (syscall 'munmap' [11])
```

`munmap`에 catchpoint를 걸어 주고 조금 더 진행해 주면..

```
   0x401611:    mov    rdi,rbp
    0x401614:    call   0x493560
    0x401619:    nop    DWORD PTR [rax+0x0]
=> 0x401620:    endbr64
   0x401624:    xor    ebp,ebp
   0x401626:    mov    r9,rdx
   0x401629:    pop    rsi
   0x40162a:    mov    rdx,rsp
```

`OEP`를 발견할 수 있다. (0x401620) 즉, 내부 언패킹 과정이 모두 끝났다는 소리이다. 지금 상태의 메모리를 덤프하면 언패킹된 바이너리를 얻을 수 있다.

```
0x00400000         0x00401000         r--p      mapped
0x00401000         0x00498000         r-xp      mapped
```

```
0x00498000         0x004c1000         r--p      mapped
0x004c1000         0x004cd000         rw-p      [heap]
```

위 메모리를 덤프하여 파일로 만들고 IDA로 플래그 출력 부분을 살펴보면 다음과 같은 코드가 존재한다.

```
v9 = __readfsqword(0x28u);
 strcpy(v8, "Y2NlMjAyMntJX0RPblRfTElrRV9Nb1pJX0JPVG5FdH0");
 printf((__int64)"flag : %d \n", 'Y');
```

패딩 기호(=)를 붙여주고 base64 decode를 진행하면 플래그를 얻을 수 있다.

```
base64_decode("Y2NlMjAyMntJX0RPblRfTElrRV9Nb1pJX0JPVG5FdH0=") = Z
```