Tuesday, January 28, 2014

PHDQuals 2014 - miXer

The task consisted solely of a 32-bit executable provided by the organizers [0]. Once we loaded it into IDA, we could see the most interesting part (main function) did not look like valid x86 code:

```
804845c:       e4 83                   in     al,0x83
804845e:       f0 55                   lock push ebp
8048460:       57                      push   edi
8048461:       e5 53                   in     eax,0x53
8048463:       81 56 89 0c 45 89 ec    adc    DWORD PTR [esi-0x77],0xec89450c
804846a:       00 01                   add    BYTE PTR [ecx],al
804846c:       8b 44 00 30             mov    eax,DWORD PTR [eax+eax*1+0x30]
8048470:       00 00                   add    BYTE PTR [eax],al
8048472:       89 24 a1                mov    DWORD PTR [ecx+eiz*4],esp
8048475:       65 00 84 14 1c c7 c0    add    BYTE PTR gs:[esp+edx*1+0x44c0c71c],al
```

Obviously, we had to fix it. Since the name of challenge was miXer, we thought we probably had to xor it with some key to obtain the original code. We could guess correct the first bytes based on the opcodes of a typical function prologue:

```
55                      push   ebp
89 e5                   mov    ebp,esp
```

Assuming the key was four bytes long, we started to brute-force the last byte... but it didn't yield any valid results. Therefore, we took a closer look at the numeric byte values and realized that the opcodes were not really modified, only shuffled around using a specific pattern: the fourth byte should be the first one, the tenth should be second and the sixth should be third. Now, we had to move the remaining bytes around in the right way to end up with proper function code.

We could make further progress by taking advantage of the fact that gcc tends to align stack to 16 bytes at the beginning of function execution with the following instruction:

```
83 e4 f0 and esp,0xfffffff0
```

At this point, we are left with four more bytes within the first 10-byte block: 53, 56, 57 and 81. The 81 byte is the first opcode of "`sub esp, xxx`" - a stack allocation, and the rest stand for "`push ebx`", "`push esi`" and "`push edi`", respectively. We could determine the order by investigating common function prologues, until we found the following one which matched:

```
55                      push   ebp
89 e5                   mov    ebp,esp
57                      push   edi
56                      push   esi
53                      push   ebx
83 e4 f0                and    esp,0xfffffff0
81 ec 30 04 00 00       sub    esp,0x430
```

If we assume that the entire function bytes are shuffled in 10-byte blocks, each in the exact same order as the first (recovered) one, we end up with the following solution in Python:

```
TEXT_START = 0x8048370
MAIN_BEG = 0x804845c - TEXT_START + 0x00370
MAIN_END = 0x8048830 - TEXT_START + 0x00370
with open('/tmp/miXer.elf.5f96dea48b8c8ab66898e902d3c98b82') as f:
  data = f.read()
code = data[MAIN_BEG:MAIN_END]
data2= list(data)
flip = lambda x: (x[1],x[0])

ctable = dict(map(flip,enumerate(code[:10])))
ctable = [ ctable[c] for c in "\x55\x89\xe5\x57\x56\x53\x83\xe4\xf0\x81"]
for i in range(len(code)/10):
  for j in range(10):
     data2[MAIN_BEG+i*10+j] = code[i*10+ctable[j]]
with open('/tmp/mixfix.bin','w') as f: f.write(''.join(data2))
```

Let's see if it works:

```
[mak@localhost tmp]$ python2 x.py
[mak@localhost tmp]$ chmod +x mixfix.bin
[mak@localhost tmp]$ ./mixfix.bin  ; echo
y0ur.f1rst.fl4g
[mak@localhost tmp]$
```


Links

Dragon Sector Web Site
[dragonsector.pl]

Contact

contact@dragonsector.pl

Archives

- ► 2017 (6)
- ► 2016 (4)
- ► 2015 (4)
- ▼ 2014 (25)
  - ► December (2)
  - ► October (1)
  - ► June (1)
  - ► May (1)
  - ► April (7)
  - ► March (4)
  - ► February (8)
  - ▼ January (1)
    - PHDQuals 2014 - miXer
- ► 2013 (22)

Popular Posts

CONFidence DS Teaser CTF registration is open!
Without further ado: http://ctf.dragonsector.pl/ . The teaser will start on the 26 th of April, 9:00 A.M. CEST (GMT+2) - for other tim...

0CTF 2017 - EasiestPrintf (PWN 150)
The task, as the name implies, was a rather basic (at first glance - there was a plot twist) format string bug in a short 32-bit Debian appl...

EKOPARTY CTF 2016 - Malware sample (RE 400)
In short, the reversing category 400 pts challenge was a journey starting with negligible x86-64 boilerplate code, leading through a somewha...

EKOPARTY CTF 2016 - FBI 100
The FBI category is something new that I personally have not seen on a CTF (though in all honesty I did have a rather long break). An FBI ta...

0CTF 2017 - complicated xss (web 177)
Complicated xss was a client-side web security task revolving around, well, XSSes. At the very start you were handed a way to XSS the adm...

0CTF 2017 - char (shellcoding 132)

Done, +2000pts. :)
[0] http://shell-storm.org/repo/CTF/PHDays-Quals-2014/MiXer-2000pts/

Posted by Maciej Kotowicz at 20:39

1 comment:

VnSpl0it 30 January, 2014 14:37

Great, Thanks !

Reply

Enter your comment...

Comment as: Select profile...

Publish Preview

Newer Post Home Older Post

The code in the "char" task was rather simple - you get to send in 2400 bytes of input (using scanf 's "%2400s", so ...

ASIS CTF Finals 2013 - Chessboard (stegano 175)

This is a task from the ASIS CTF Finals 2013, "Stego" (steganography) category, and it was solved by 2 teams including ours. This ...

0CTF 2017 - UploadCenter (PWN 523)

Welcome to another Menu Chall right~ Here you can use any function as you wish No more words , Let't begin 1 :) Fill your information...

UFO CTF 2013 - Broken brokoli (forensics 100)

This is a task from UFO CTF 2013, which was a sweet mixture of file format stegano, forensics and decoding weird alphabets (though that'...

Nuit du Hack CTF Quals 2014 - Nibble

The recent Nuit Du Hack CTF Quals CTF was mostly web, crypto and forensics-oriented, with no tasks explicitly categorized as "Exploita...

Contributors

- Adam Iwaniuk
- Gynvael Coldwind
- Jagger
- Krzysztof Katowicz-Kowalewski
- Lympho Cytus
- Marcin Kalinowski
- Mateusz P
- Michał Kowalczyk
- Tomasz Bukowski
- Tomasz Dubrownik
- j00ru
- q3k
- valis

Mathjax