

PWVN题的逆向分析策略

讲师：Atum

内容简介

INTRODUCTION

01 逆向工程简介

02 常见漏洞简介

03 PWN题的逆向分析技巧

逆向工程简介

从目标代码反推源代码的过程叫做逆向

对于PWN 来说，逆向工程的主要作用为发掘与分析漏洞

工具：

- 静态分析工具：IDA pro
- 动态调试工具：gdb，windbg，ollydbg，IDA pro

IDA pro的F5大法

```
1 signed __int64 __fastcall toHex(unsigned __int8 i)
2 {
3     char *p; // [rsp+10h] [rbp-10h]@1
4
5     p = strchr("0123456789ABCDEF", i);
6     if ( !p )
7         dead();
8     return p - "0123456789ABCDEF";
9 }
```

```
.text:000000000400D2C ; unsigned __int8 __cdecl toHex(unsigned __int8 i)
.text:000000000400D2C public toHex
.text:000000000400D2C toHex proc near ; CODE XREF: hextobin
.text:000000000400D2C ; hextobin+6C↓p
.text:000000000400D2C i = byte ptr -14h
.text:000000000400D2C p = qword ptr -10h
.text:000000000400D2C hexTable = qword ptr -8
.text:000000000400D2C
.text:000000000400D2C push rbp
.text:000000000400D2D mov rbp, rsp
.text:000000000400D30 sub rsp, 20h
.text:000000000400D34 mov eax, edi
.text:000000000400D36 mov [rbp+i], al
.text:000000000400D39 mov [rbp+hexTable], offset a0123456789abc
.text:000000000400D41 movzx edx, [rbp+i]
.text:000000000400D45 mov rax, [rbp+hexTable]
.text:000000000400D49 mov esi, edx ; c
.text:000000000400D4B mov rdi, rax ; s
.text:000000000400D4E call _strchr
.text:000000000400D53 mov [rbp+p], rax
.text:000000000400D57 cmp [rbp+p], 0
.text:000000000400D5C jz short loc_400D6E
.text:000000000400D5E mov rdx, [rbp+p]
.text:000000000400D62 mov rax, [rbp+hexTable]
.text:000000000400D66 sub rdx, rax
.text:000000000400D69 mov rax, rdx
.text:000000000400D6C jmp short locret_400D73
.text:000000000400D6E ; -----
.text:000000000400D6E loc_400D6E: call dead ; CODE XREF: toHex+30
.text:000000000400D73 ; -----
.text:000000000400D73 locret_400D73: ; CODE XREF: toHex+40
.text:000000000400D73 leave
.text:000000000400D74 retn
.text:000000000400D74 toHex endp
```

常见漏洞简介

在进行漏洞挖掘之前，必须对常见漏洞非常熟悉！！！！

- 建议先在Wiki了解以下各个漏洞的概念和利用方式，然后再根据下面列出的题目来看看这些漏洞在实际题目中的样子

缓冲区溢出 (Buffer Overflow)

- 堆溢出、栈溢出、bss溢出、data溢出 (通常覆盖指针)
- wellpwn , AliCTF 2016 vss , Hitcon 2015 readable , stkof , zerostorage

整数溢出 (Integer Overflow)

- 无符号型与有符号的转换 (MMACTF 2016 shadow)
- 整数加减乘法，如malloc(size*2) (pwnhub.cn calc)
- 整数溢出通常会进一步转换为缓冲区溢出、逻辑漏洞等其他漏洞

常见漏洞简介

格式化字符串 (Format String)

- printf(s),sprintf(s),fprintf(s)等，可能导致任意地址读写 (MMACTF 2016 greeting)
- 可以用来leak(HCTF2016 fheap)

释放后使用 (Use-After-Free)

- 释放掉的内存可能会被重新分配，释放后使用会导致重新分配的内存被旧的使用所改写
- Double free是一种特殊的UAF
- Defcon 2014 Qualifier shitsco , AliCTF 2016 router , OCTF2016 freenote (double free) , HCTF2016 fheap (double free)

逻辑漏洞

- 访问控制，协议漏洞，多线程竞态条件(fakefuzz)等

漏洞挖掘中的逆向技巧

关键数据结构分析：还原结构体、接口、类等。

控制流分析：理清楚程序的执行逻辑，基本要做到从反汇编代码到源码的还原。

数据流分析：理清楚数据的流向。

CTF漏洞挖掘中的分析策略：

- 目标文件较小时，通常采用对整个目标文件进行控制流分析，做到整个程序从反汇编代码到接近源码级别的还原，还原的同时查找漏洞
- 目标文件较大时，逆向整个文件所需工作量太大，通常需要额外的关注数据流，并理清楚数据流所经之处的控制流，因为漏洞的触发与数据流离不开关系
- 无论是数据流分析和控制流分析，还原结构体、接口、类都会促进逆向工程

关键数据结构分析

在分析控制流的时，根据程序对内存块的操作方法，还原出结构体出结构体

- 不一定要完全还原，对于不知道的域可以先填写unknow，还原结构体可方法对控制流的分析

```
__int64 do_dump()
{
    __int64 result; // rax@1
    __int64 node; // rbx@1
    __int64 rightnode; // rax@5

    puts("INFO: Dumping all rows.");
    result = (__int64)&rowkey;
    node = *(_QWORD *)&rowkey;
    if ( *(_QWORD *)&rowkey )
    {
        while ( *(_QWORD *)(node + 24) )
            node = *(_QWORD *)(node + 24);
        while ( 1 )
        {
            while ( 1 )
            {
                *(_QWORD *)(node + 8);
                __printf_chk(1LL, "INFO: Row [%s], %zd byte%s\n", *(_QWORD *)node);
                rightnode = *(_QWORD *)(node + 32);
                if ( !rightnode )
                    break;
                do
                {
                    node = rightnode;
                    rightnode = *(_QWORD *)(rightnode + 24);
                }
                while ( rightnode );
            }
            result = *(_QWORD *)(node + 40);
        }
    }
}
```

tree	struct ;
key	dq ?
size	dq ?
data	dq ?
left	dq ?
right	dq ?
father	dq ?
unknow	dq ?
tree	ends

```
tree *do_dump()
{
    tree *result; // rax@1
    tree *node; // rbx@1
    tree *rightnode; // rax@5

    puts("INFO: Dumping all rows.");
    result = (tree *)&rowkey;
    node = *(tree **)&rowkey;
    if ( *(_QWORD *)&rowkey )
    {
        while ( node->left )
            node = (tree *)node->left;
        while ( 1 )
        {
            while ( 1 )
            {
                node->size;
                __printf_chk(1LL, "INFO: Row [%s], %zd byte%s\n", node->key);
                rightnode = (tree *)node->right;
                if ( !rightnode )
                    break;
                do
                {
                    node = rightnode;
                    rightnode = (tree *)rightnode->left;
                }
                while ( rightnode );
            }
            result = (tree *)node->father;
        }
    }
}
```

控制流分析

控制流分析的主要作用是理清楚程序的逻辑，对于规模较小的目标文件，一般选择理清整个目标文件。

代码以识别为主，不要硬逆。

- 需要熟悉常见的数据结构、算法在目标文件一般“长啥样”
 - 链表、树、图、堆、各种加密算法等
- 逆向分析是一个经验性的工作，刚开始慢慢来，逆多了自会有所感悟
- 善用标记，标记结构体、标记变量名、标记变量类型
- F5大法好，但是F5不是万能的，当发现F5结果比较诡异时需要在汇编层分析（如mmactf2016 shadow）
- 再次强调，需要熟悉各类漏洞，否则碰到漏洞也不知道是漏洞

数据流分析

目标文件较大，全盘逆向不现实

- 追溯用户输入的走向，重点关注对用户输入数据处理的函数
- 可以在不用逆清楚控制流即可找到漏洞，需要一定的技巧性

plaidCTF 2015 datastore

The image features a white background with two large yellow geometric shapes in the corners. The shape in the top-left corner is a triangle pointing towards the bottom-right. The shape in the bottom-right corner is a more complex polygon, also pointing towards the top-left, and it overlaps with a lighter yellow shape underneath it.

The End