CTF中的其他漏洞

讲师：Atum

内容简介

INTRODUCTION

01 格式化字符串

02 竞争条件漏洞

03 代码逻辑漏洞

04 类型混淆漏洞

05 缓冲区未初始化

格式化字符串漏洞

格式化串介绍：https://en.wikipedia.org/wiki/Printf_format_string

格式化串相关函数：printf、fprintf、sprintf等

格式化漏洞：

- 触发形如printf(fmtstr)的调用，fmtstr=”%$22p”时可以打印栈指针之后第22个 DWORD, fmtstr=“%1000c%$22n”时可以将栈指针之后第22个DWORD作为地址写入1000
- https://www.exploit-db.com/docs/28476.pdf

例题

- MMACTF 2016 greeting
- HCTF 2016 fheap (利用格式化串漏洞leak栈中的数据)
- RuCTF 2016 weather
- **作业**：完成MMACTF 2016 greeting

竞争条件漏洞

竞争条件：竞争条件指多个线程或者进程在读写一个共享数据。竞争条件发生时程序最终的结果依赖于多个进程的指令执行顺序

通过竞争条件，我们可以让程序执行超出预期的行为，如：

- Thread1: free(p1), p1=0;
- Thread2: p2=malloc(), read(p2), p1->callfunc()

如果Thread1执行完第一条语句，Thread2开始执行并执行至最后一条语句，则会产生Use After Free 进而控制流劫持。

关键：如何控制线程执行的顺序。大名鼎鼎的脏牛就是这类漏洞

例题：

- 安恒杯 武汉大学邀请赛 fackfuzz (多线程竞争条件造成UAF)
- Stupid shell (多进程竞争条件造成越权读文件)

代码逻辑漏洞

代码在实现过程中因逻辑错误所产生的漏洞，如：

```
if (len>=size)  free(ptr);
if (len>size) ptr=malloc(len), read(ptr);
else read(ptr)
```

第二个if少了一个等号，从而导致了UAF。

代码逻辑漏洞的表现方式多种多样，没有定式，难以自动化检测

例题：UCTF 2016 note 代码逻辑漏洞导致的控制流劫持

类型混淆漏洞

将类型A的对象当作类型B的对象进行引用与操作

- 显式类型转换导致的类型混淆：static_cast 子类转父类
  - static_cast在编译时C++编译器会检查转换前的类型和转换后的类型是否兼容，编译器只允许derived class转 换为base class 或 base class转换为derived class。前者称为upcast, 后者称为downcast。由于derived class的内存布局通常包含base class，所以upcast通常 会存在安全问题，但是downcast却可能会存在类型混淆的问题。 如说在base class的对象中使 个只有derived class存在的成员就会产 内存越界访问
- 代码中的逻辑错误或漏洞导致的类型混淆 ：破坏了类/结构体中的表示类型数据（广义）
  - 如修改vatable也算一种类型混淆

例题：无，建议看看类型混淆的CVE，如CVE-2015-3077，目前在CTF中没有看到相关题目

缓冲区未初始化漏洞

栈未初始化时，栈中数据为上次函数调用留下的栈帧

堆未初始化时，堆中数据为上次使用该堆块所留下的数据

利用举例：

- 若data在栈上且未初始化，代码read(data);puts(data);可能会造成栈中数据被打印出来
- 若ptr在栈上且未初始化，代码read(ptr);可能会导致任意写
- 若ptr在堆上且未初始化,if(ptr->data!=null) read(ptr->data) 可能导致任意写

例题：

UCTF 2016 note （可以用栈未初始化漏洞做)

华山杯2016决赛 SU_PWN （栈未初始化漏洞泄漏栈地址）

33C3 CTF PWN

The End